MedTrack: aws cloud-enabled healthcare management system

Project Description:

In today’s fast-evolving healthcare landscape, ef icient communication and coordination between doctors and patients are crucial. MedTrack is a cloud-based healthcare management system that streamlines patient doctor interactions by providing a centralized platform for booking appointments, managing medical histories, and enabling diagnosis submissions. To address these challenges, the project utilizes Flask for backend development, AWS EC2 for hosting, and DynamoDB for managing data. MedTrack allows patients to register, log in, book appointments, and submit diagnosis reports online. The system ensures real-time noti ications, enhancing communication between doctors and patients regarding appointments and medical submissions. Additionally, AWS Identity and Access Management (IAM) is employed to ensure secure access control to AWS resources, allowing only authorized users to access sensitive data. This cloud-based solution improves accessibility and ef iciency in healthcare services for all users.

Scenarios:

Scenario 1: Ef icient Appointment Booking System for Patients

In the MedTrack system, AWS EC2 provides a reliable infrastructure to manage multiple patients accessing the platform simultaneously. For example, a patient can log in, navigate to the appointment booking page, and easily submit a request for an appointment. Flask handles backend operations, ef iciently retrieving and processing user data in real-time. The cloud-based architecture allows the platform to handle a high volume of appointment requests during peak periods, ensuring smooth operation without delays.

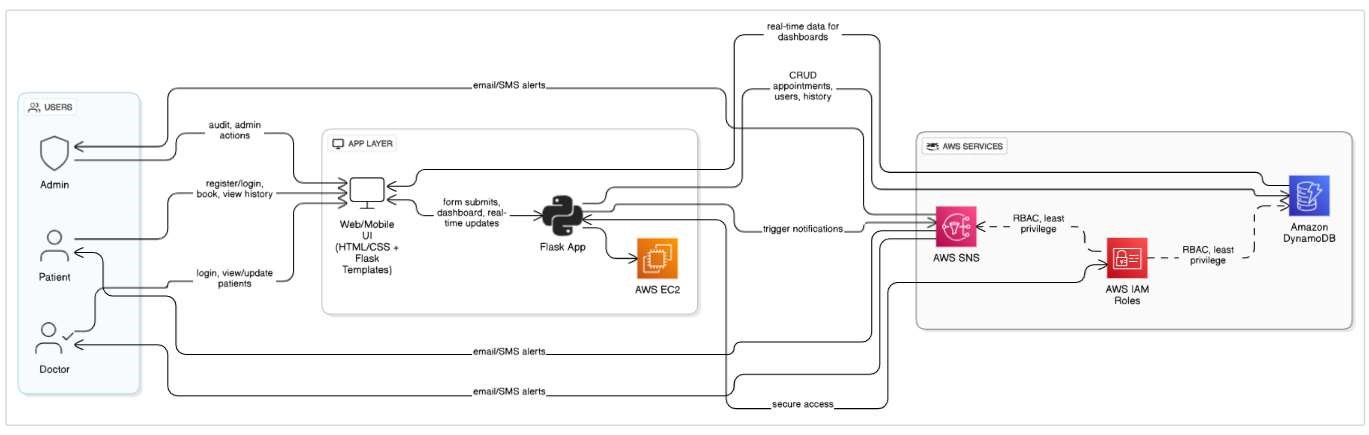
Scenario 2: Secure User Management with IAM

MedTrack utilizes AWS IAM to manage user permissions and ensure secure access to the system. For instance, when a new patient registers, an IAM user is created with speci ic roles and permissions to access only the features relevant to them. Doctors have their own IAM con igurations, allowing them access to patient records and appointment details while maintaining strict security protocols. This setup ensures that sensitive data is accessible only to authorized users.

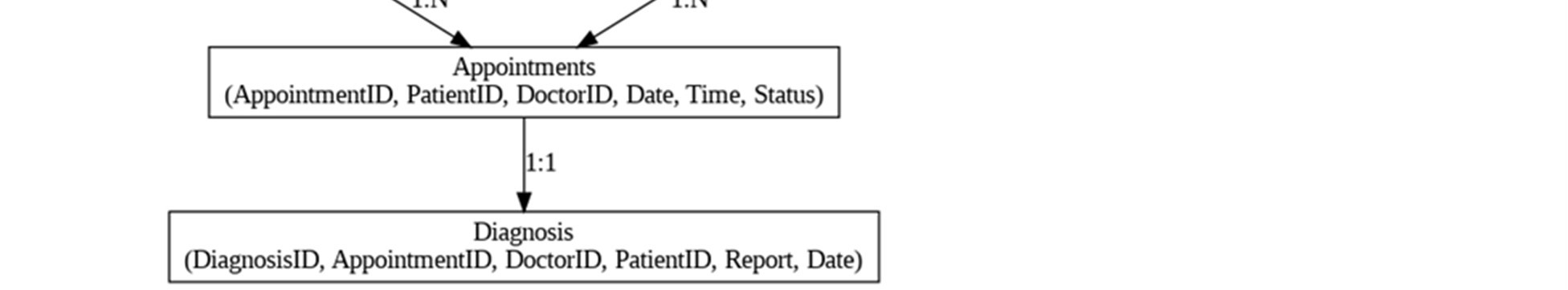
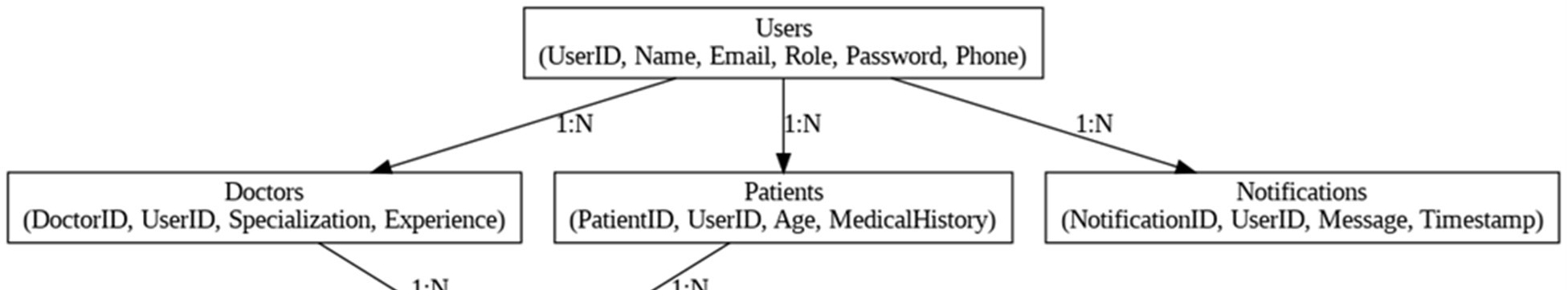
Scenario 3: Easy Access to Medical History and Resources

The MedTrack system provides doctors and patients with easy access to medical histories and relevant resources. For example, a doctor logs in to view a patient's medical history and upcoming appointments. They can quickly access, and update records as needed. Flask manages real-time data fetching from DynamoDB, while EC2 hosting ensures the platform performs seamlessly even when multiple users access it simultaneously, offering a smooth and uninterrupted user experience.

AWS ARCHITECTURE :



Entity Relationship (ER)Diagram:



Pre-requisites:

1. .AWS Account Setup: AWS Account Setup
2. Understanding IAM: IAM Overview
3. Amazon EC2 Basics: EC2 Tutorial
4. DynamoDB Basics: DynamoDB Introduction
5. SNS Overview: SNS Documentation
6. Git Version Control: Git Documentation

# Project WorkFlow

Milestone 1. Web Application Development and SetupDevelop the Backend Using Flask.  Integrate AWS Services Using boto3.

Milestone 2. AWS Account Setup and Login

* Set up an AWS account if not already done.
* Login to AWS Management Console.

Milestone 3. DynamoDB Database Creation and Setup

* Create a DynamoDB Table.
* Con igure Attributes for User Data and Book Requests.

Milestone 4. SNS Noti ication Setup

* Create SNS topics for book request noti ications.
* Subscribe users and library staff to SNS email noti ications.

Milestone 5. IAM Role Setup

* Create IAM Role
* Attach Policies

Milestone 6. EC2 Instance Setup

* Launch an EC2 instance to host the Flask application.
* Con igure security groups for HTTP, and SSH access.

Milestone 7. Deployment using EC2

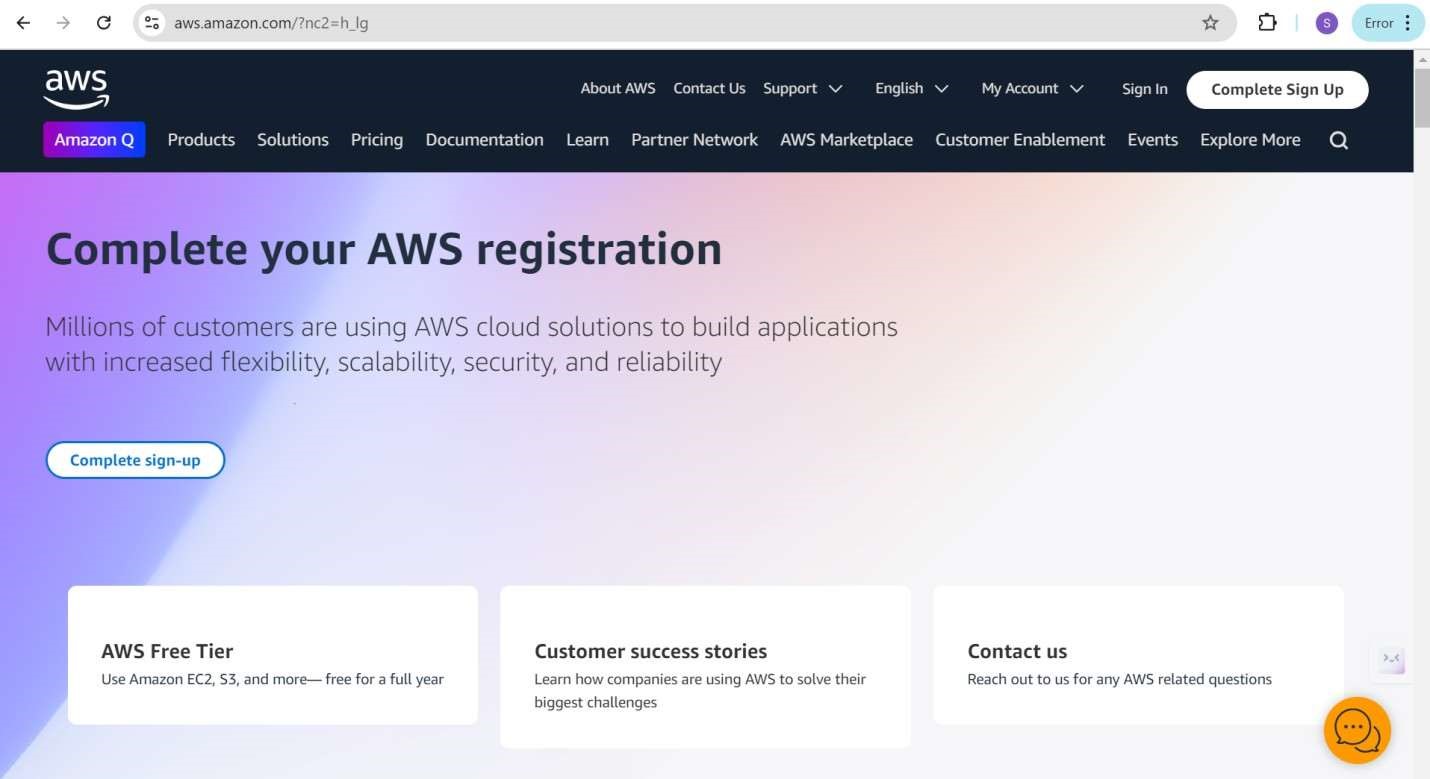
* Upload Flask Files
* Run the Flask App

Milestone 8. Testing and Deployment

* Conduct functional testing to verify user registration, login, book requests, and noti ications.

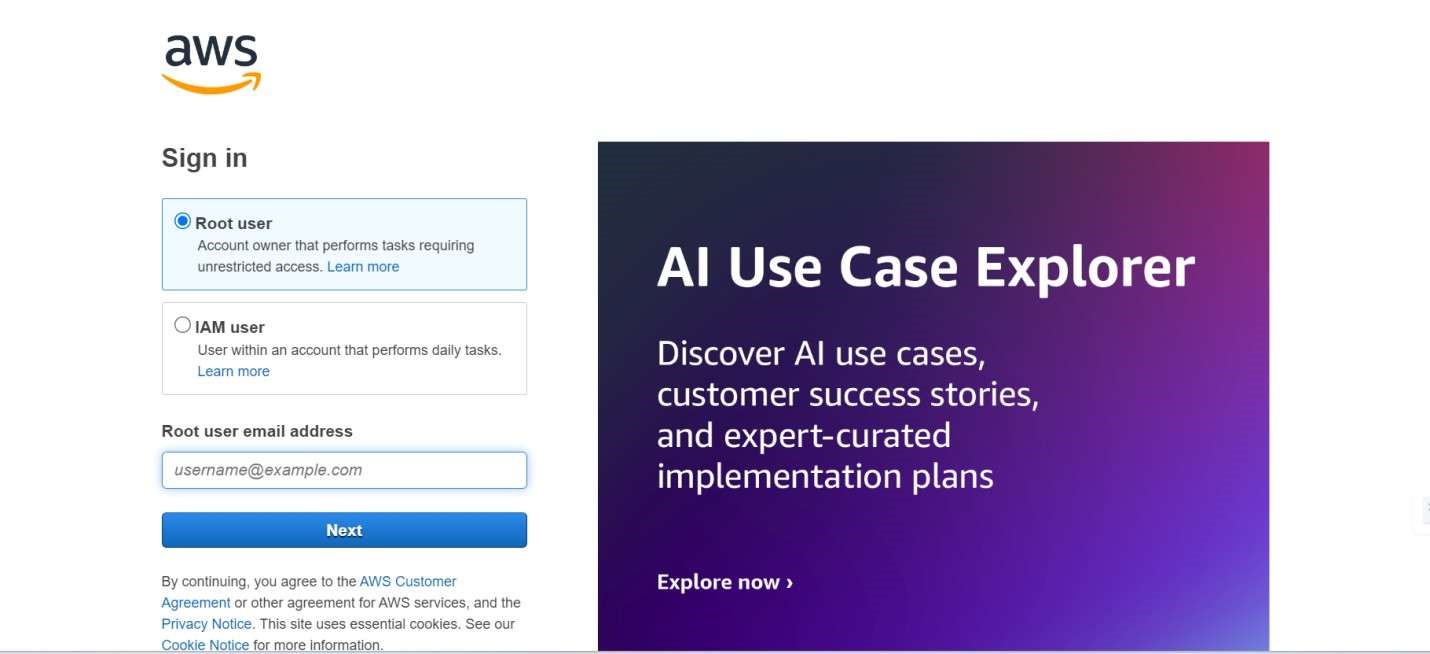
Milestone 1: Web Application Development and Setup

* Activity 1.1: Set up an AWS account if not already done.
  1. Sign up for an AWS account and con igure billing settings.



* Activity 1.2: Log in to the AWS Management Console

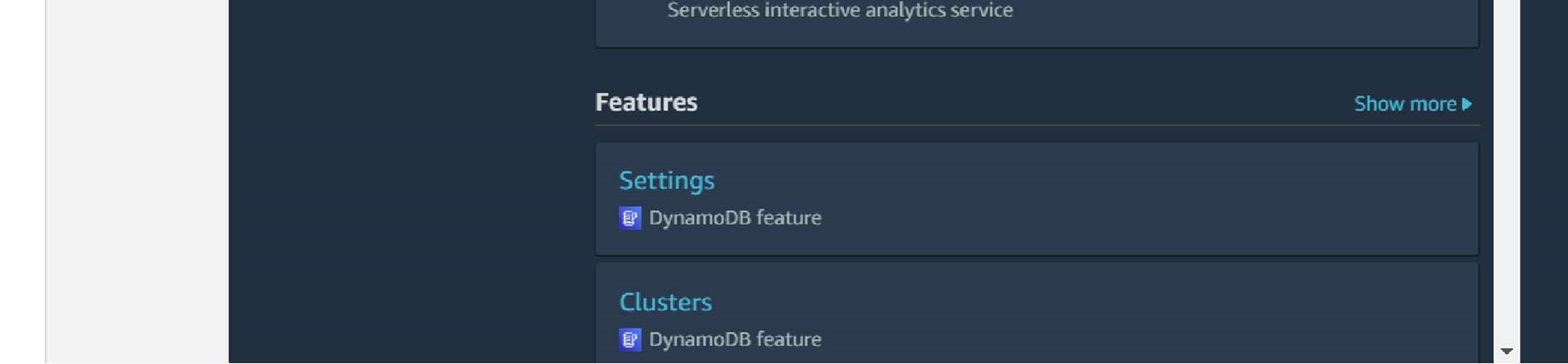
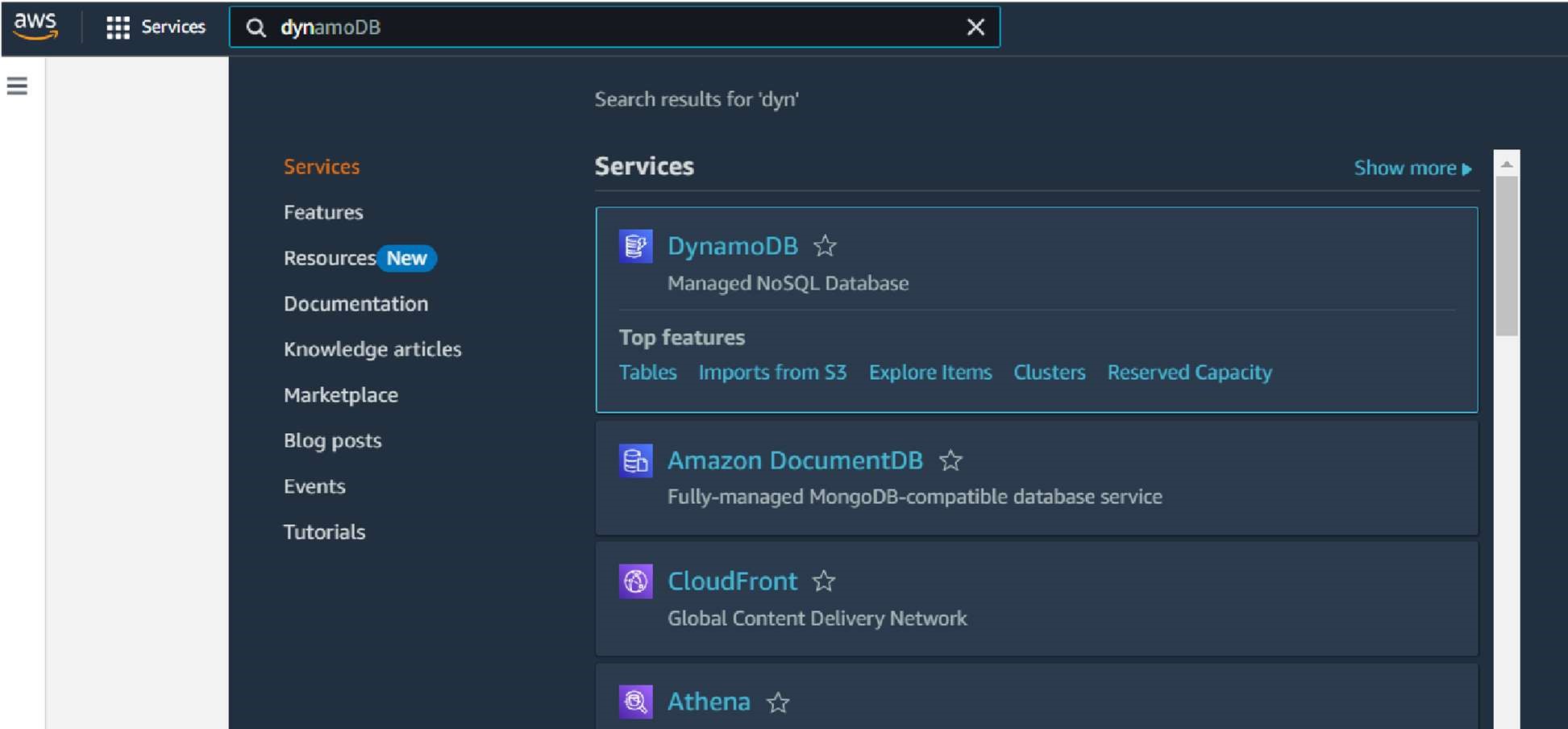
* 1. After setting up your account, log in to the AWS Management Console.



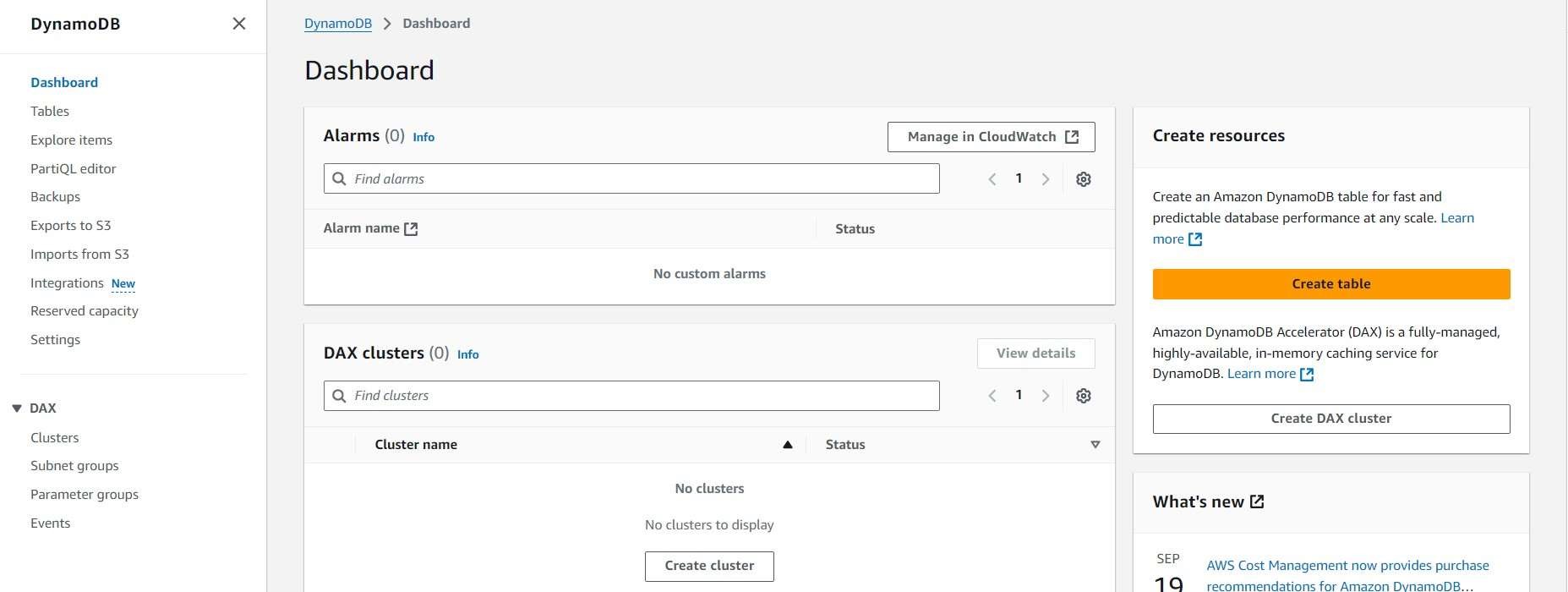
# Milestone 2: DynamoDB Database Creation and Setup

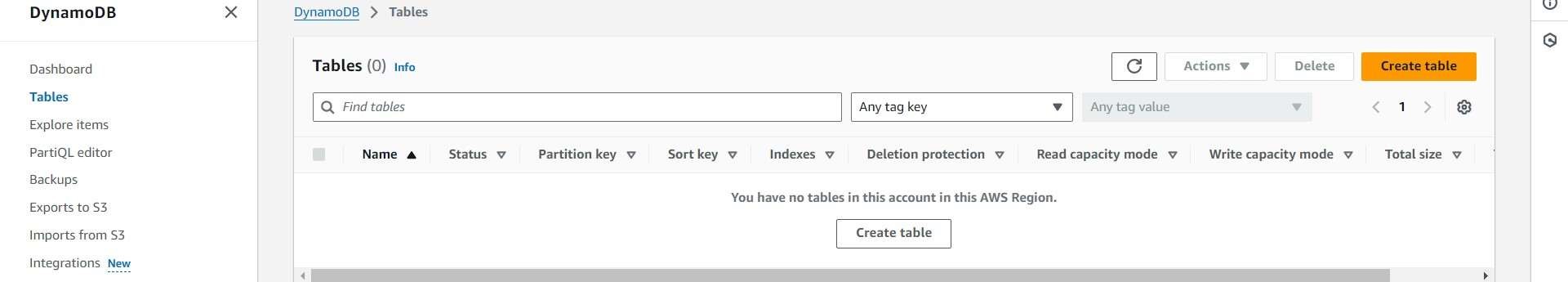
* Activity 2.1:Navigate to the DynamoDB

 In the AWS Console, navigate to DynamoDB and click on create tables.



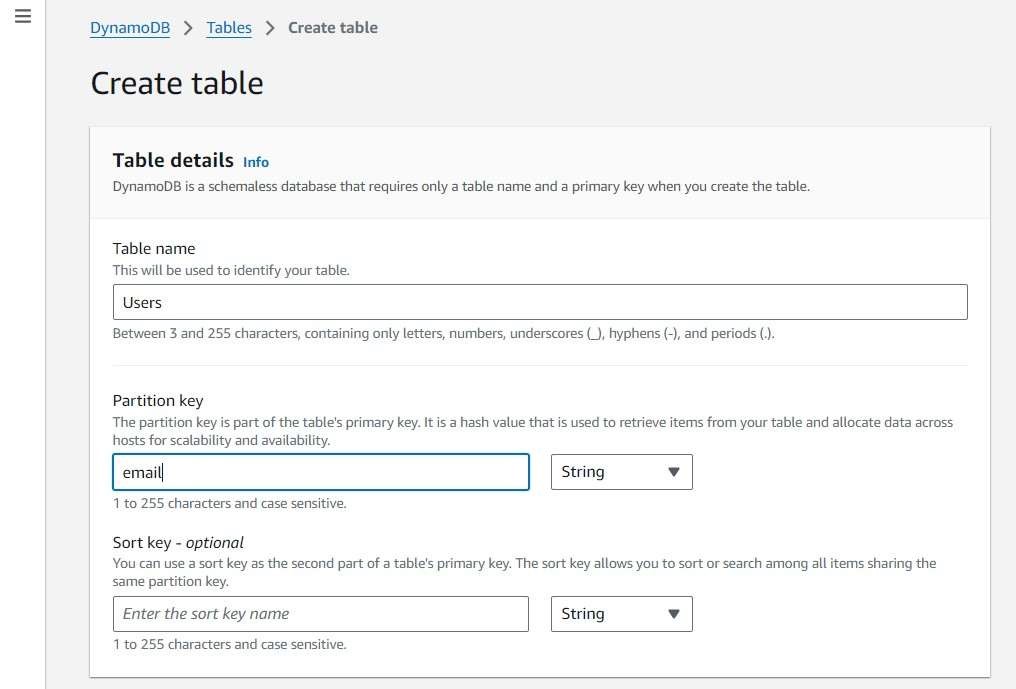
○

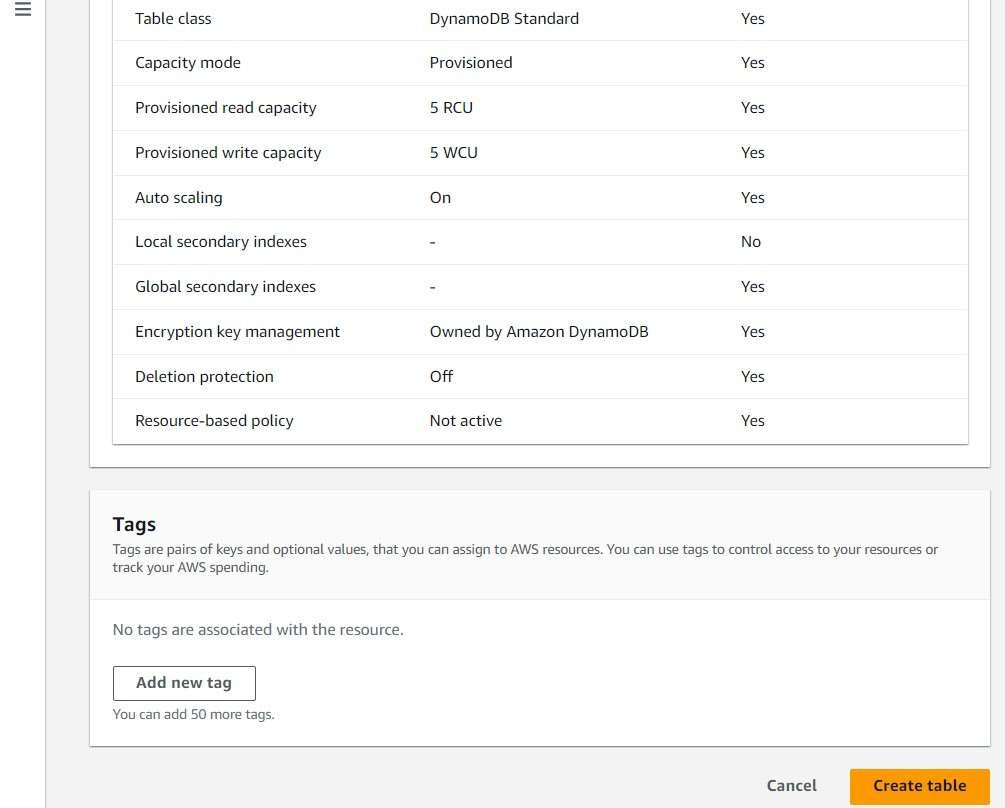


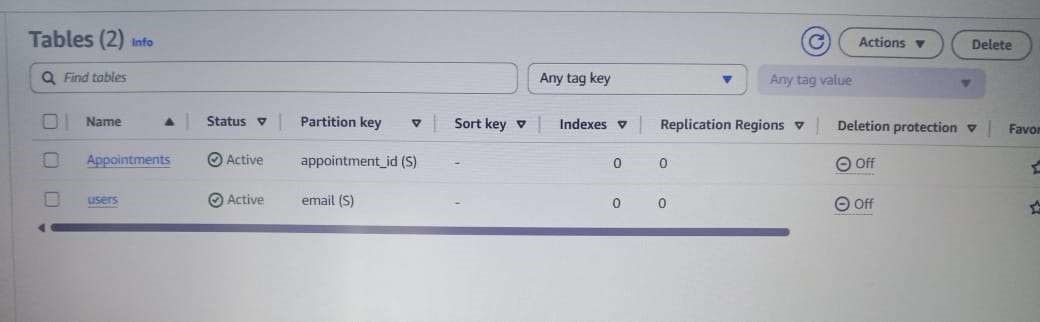


* Activity 2.2:Create a DynamoDB table for storing registration details and book requests.

 Create Users table with partition key “Email” with type String and click on create tables.





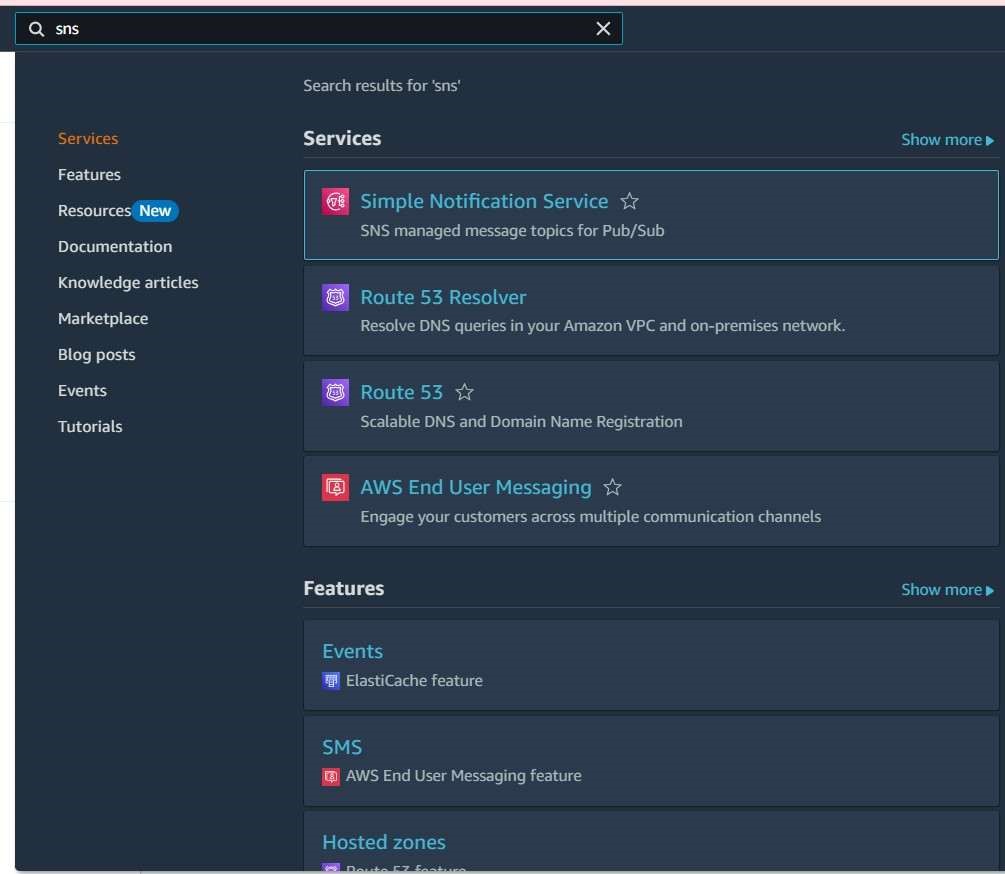


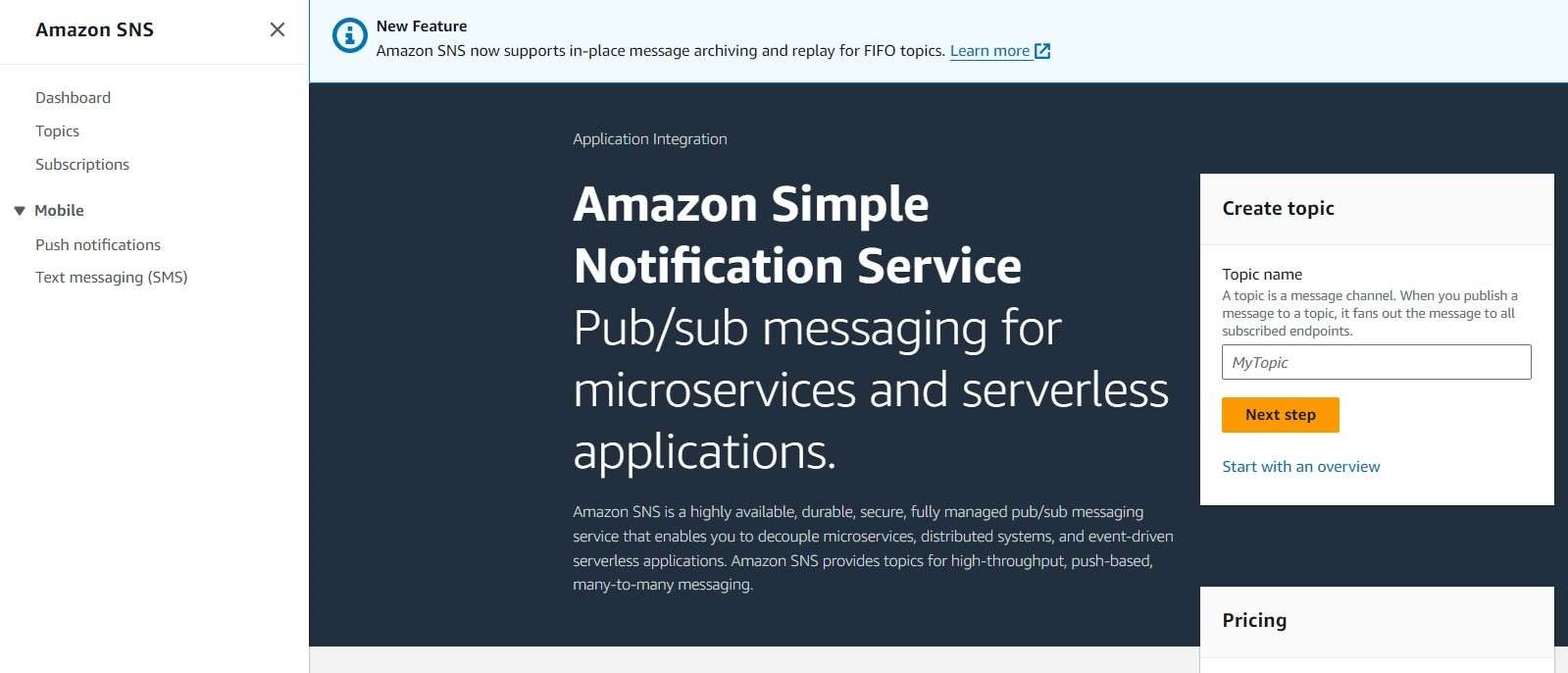
○ Follow the same steps to create a requests table with Email as the primary key for book requests data.

## Milestone 3: SNS Noti ication Setup

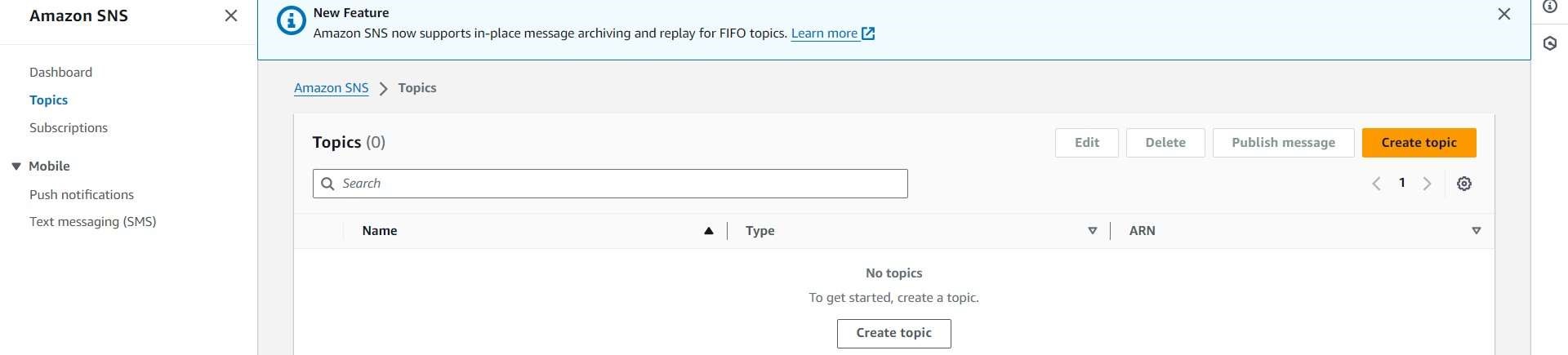
● Activity 3.1: Create SNS topics for sending email noti ications to users and library staff.

 In the AWS Console, search for SNS and navigate to the SNS Dashboard.

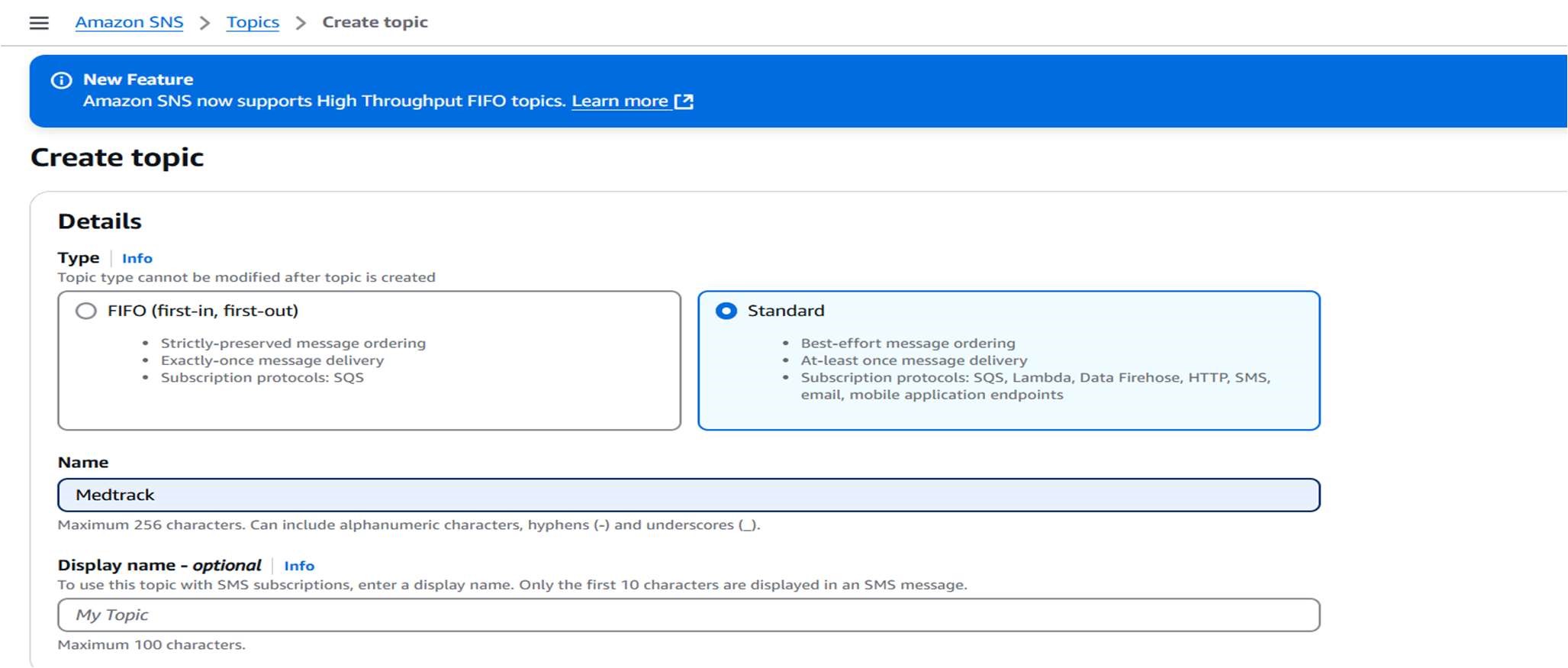


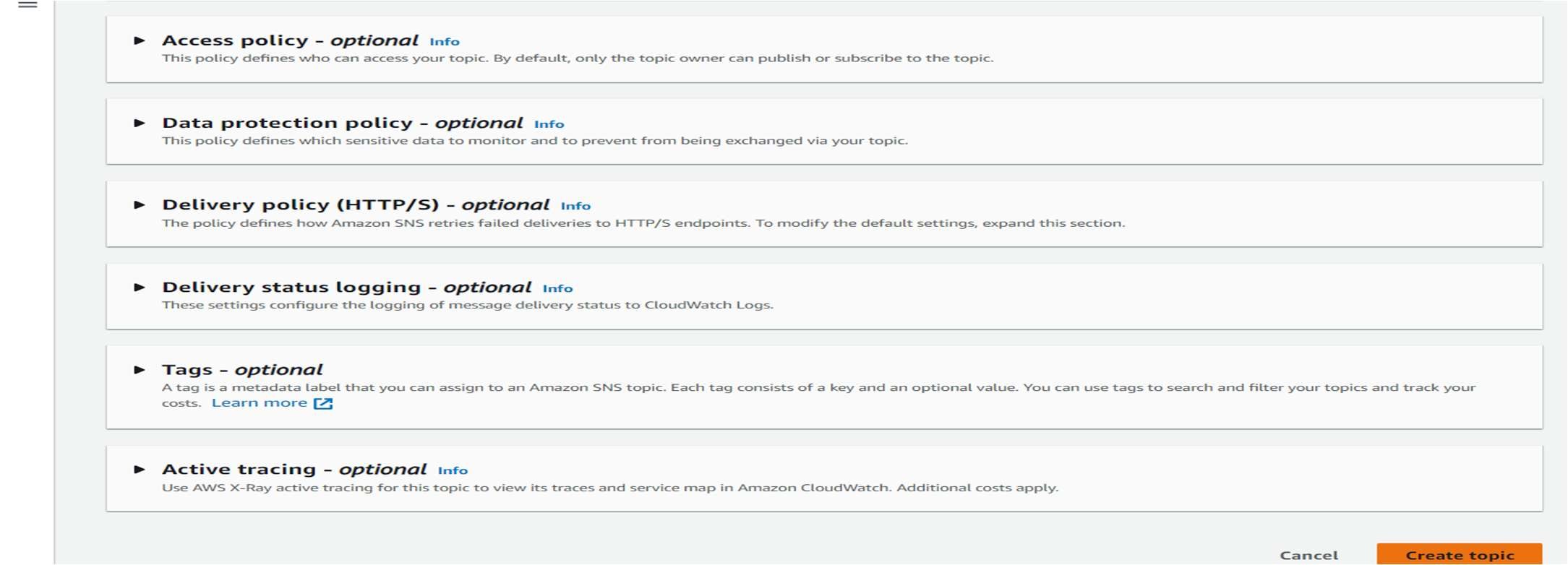


○ Click on Create Topic and choose a name for the topic.

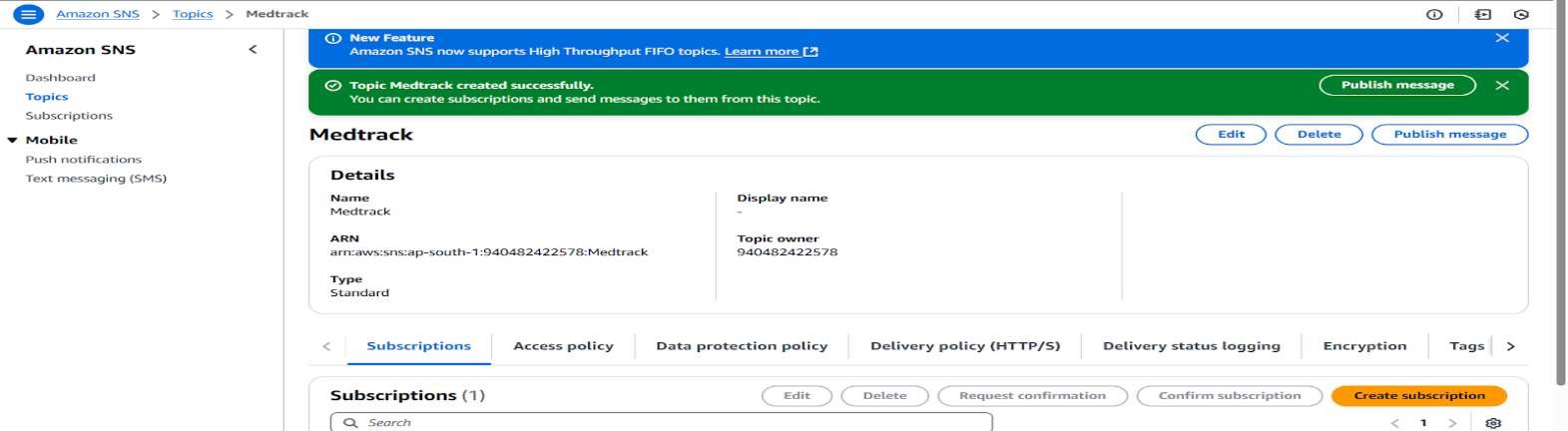


○ Choose Standard type for general noti ication use cases and Click on Create Topic.





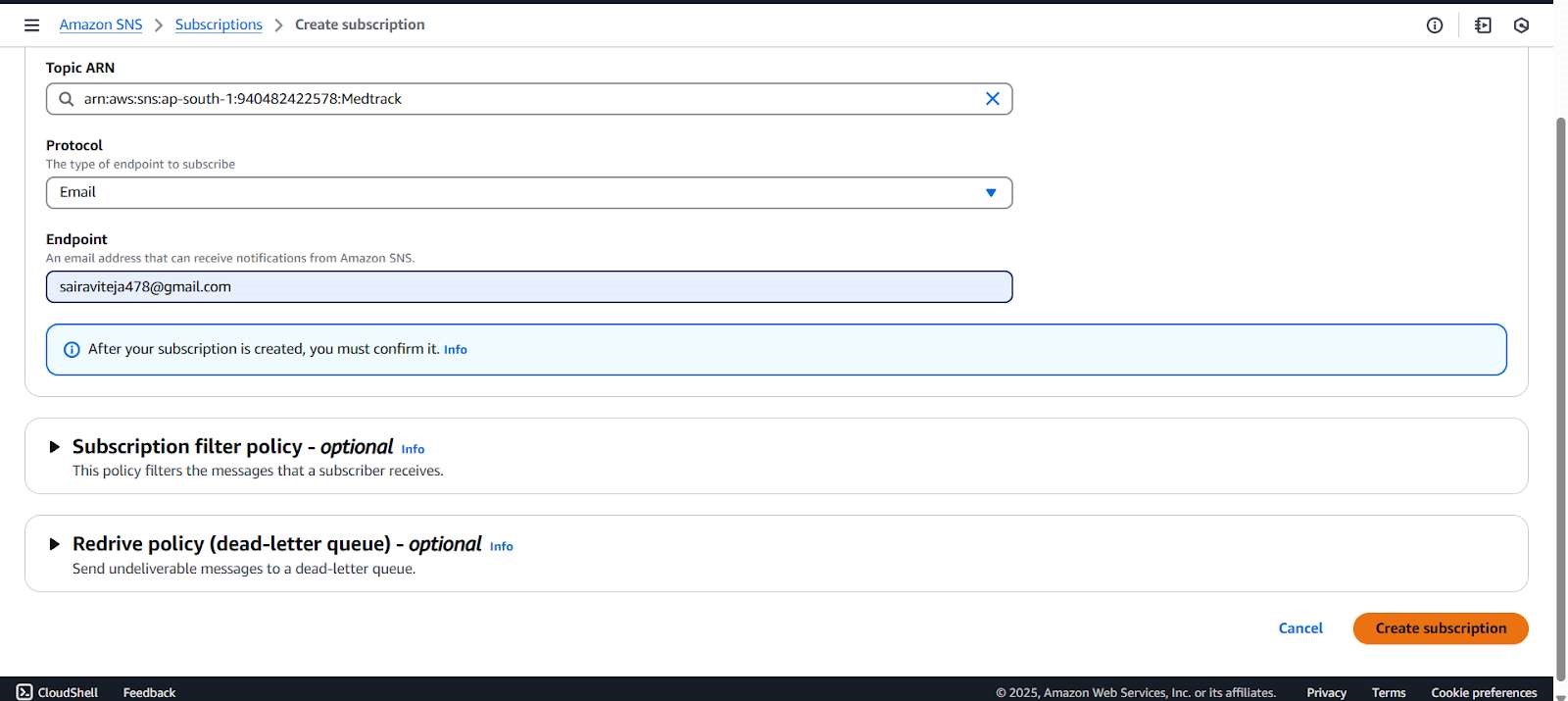
○ Con igure the SNS topic and note down the Topic ARN.



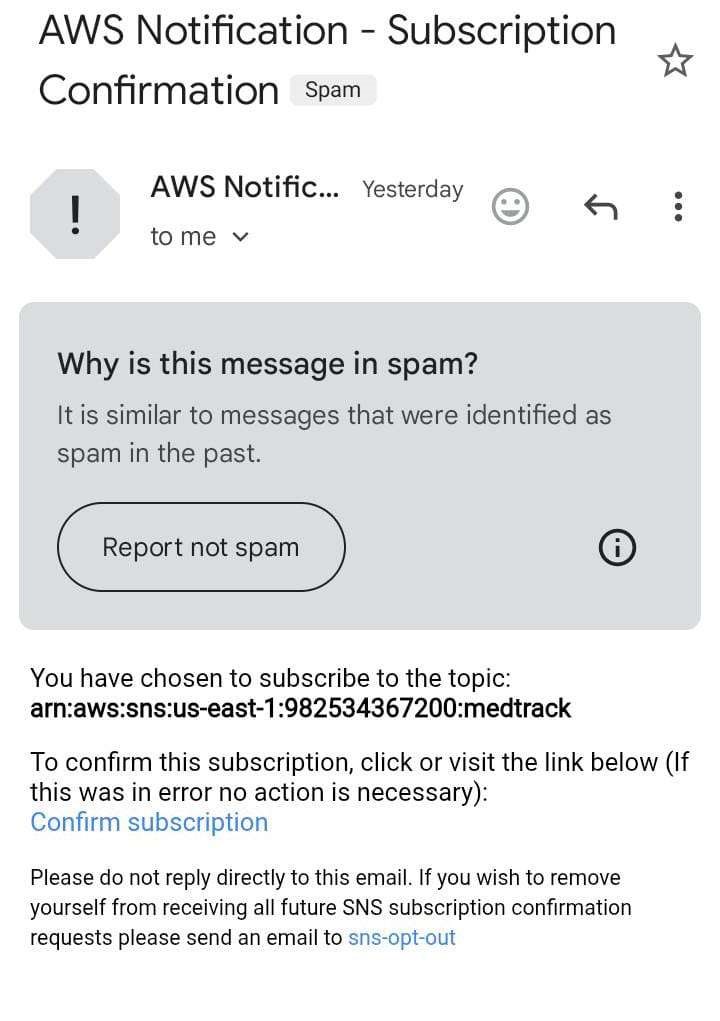
.Activity 3.2: Subscribe users and staff to relevant SNS

topics to receive real-time noti ications when a book request is made.

 Subscribe users (or admin staff) to this topic via Email. When a book request is made, noti ications will be sent to the subscribed emails.



○ After subscription request for the mail con irmation



○ Navigate to the subscribed Email account and Click on the con irm subscription in the AWS Noti ication- Subscription Con irmation mail.

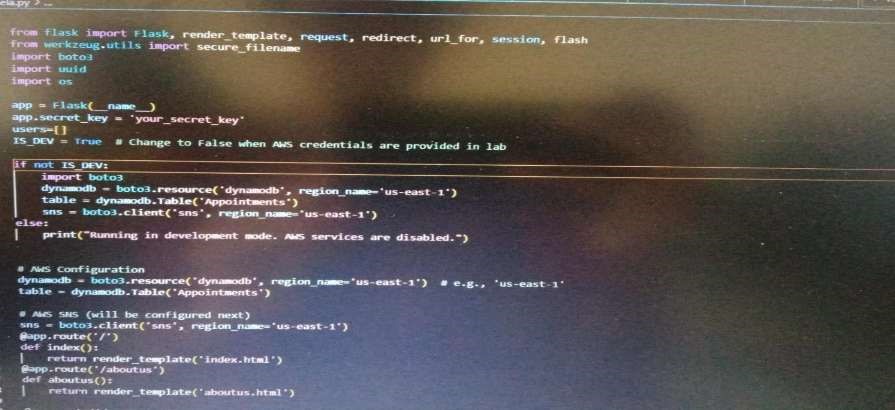


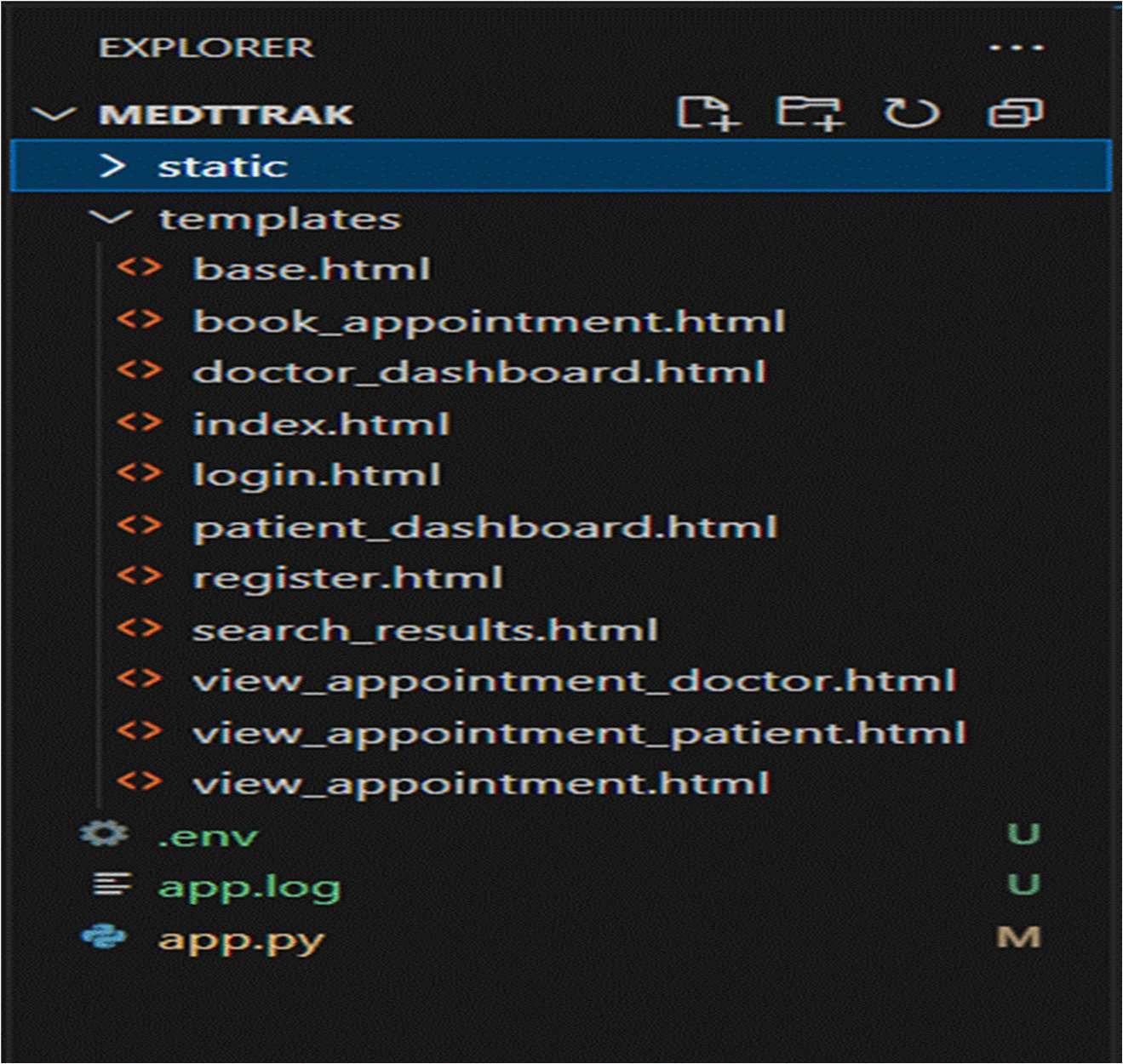
○ Successfully done with the SNS mail subscription and setup, now store the ARN link.

# Milestone 4:Backend Development and Application Setup

* Activity 4.1: Develop the backend using Flask

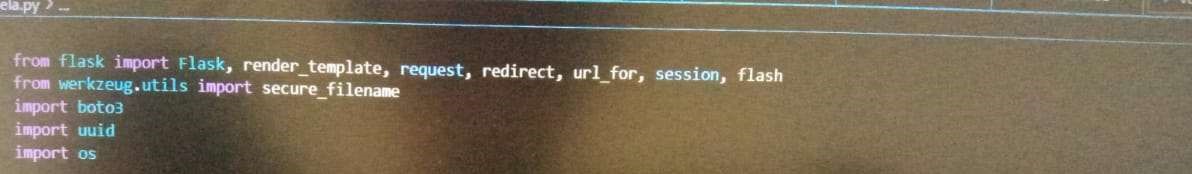
○ File Explorer Structure





Description: set up the INSTANT LIBRARY project with an app.py ile, a static/ folder for assets, and a templates/ directory containing all required HTML pages like home, login, register, subject-speci ic pages (e.g., computer\_science.html, data\_science.html), and utility pages (e.g., request-form.html, statistics.html).

Description of the code :



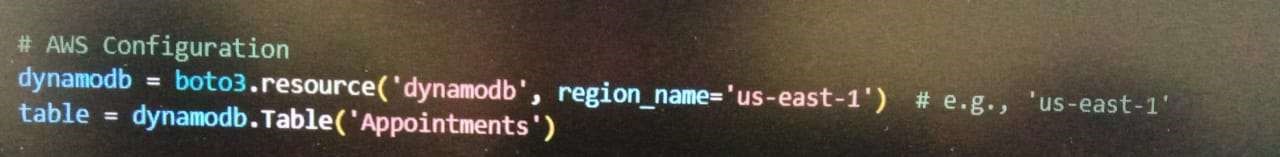
* Flask App Initialization

Description: import essential libraries including Flask utilities for routing, Boto3 for DynamoDB operations, SMTP and email modules for sending mails, and Bcrypt for password hashing and veri ication



Description: initialize the Flask application instance using Flask(\_\_name\_\_) to start building the web app.

* Dynamodb Setup:



Description: initialize the DynamoDB resource for the ap-south-1 region and set up access to the Users and Requests tables for storing user details and book requests.

* SNS Connection



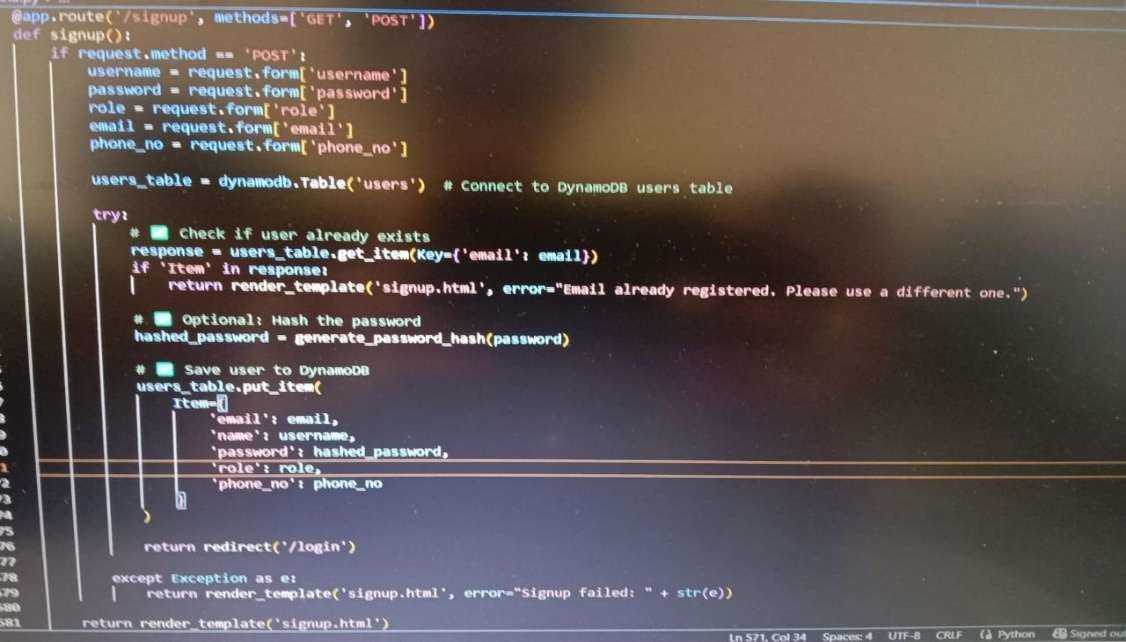
Description: Con igure SNS to send noti ications when a book request is submitted. Paste your stored ARN link in the sns\_topic\_arn space, along with the region name where the SNS topic is created. Also, specify the chosen email service in SMTP\_SERVER (e.g., Gmail, Yahoo, etc.) and enter the subscribed email in the SENDER\_EMAIL section. Create an ‘App password’ for the email ID and store it in the SENDER\_PASSWORD section.

* Routes for Web Pages
* index Route:



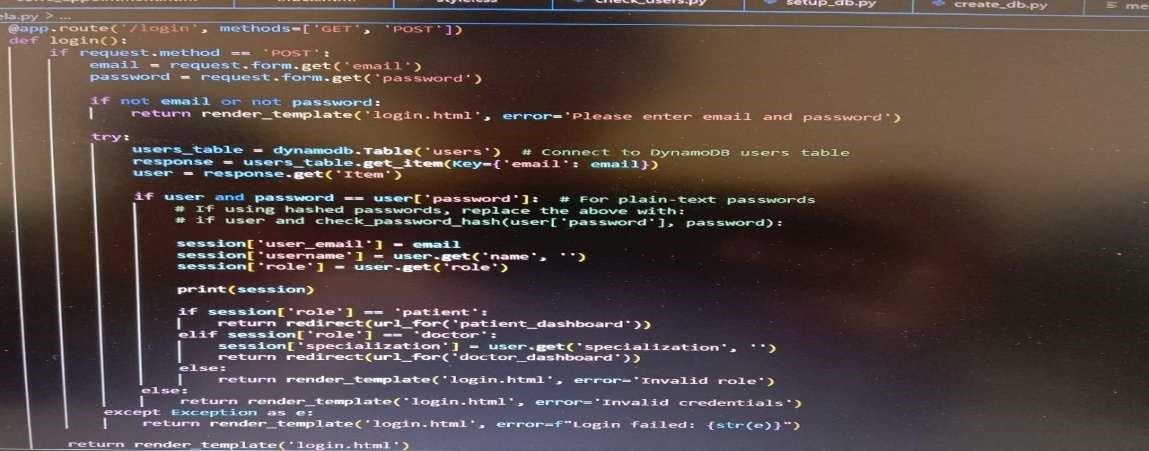
Description: de ine the index route / to automatically redirect users to the signup or login or aboutus or contactus page based on button we have chosen when they access the base URL.

 Sign up Route:



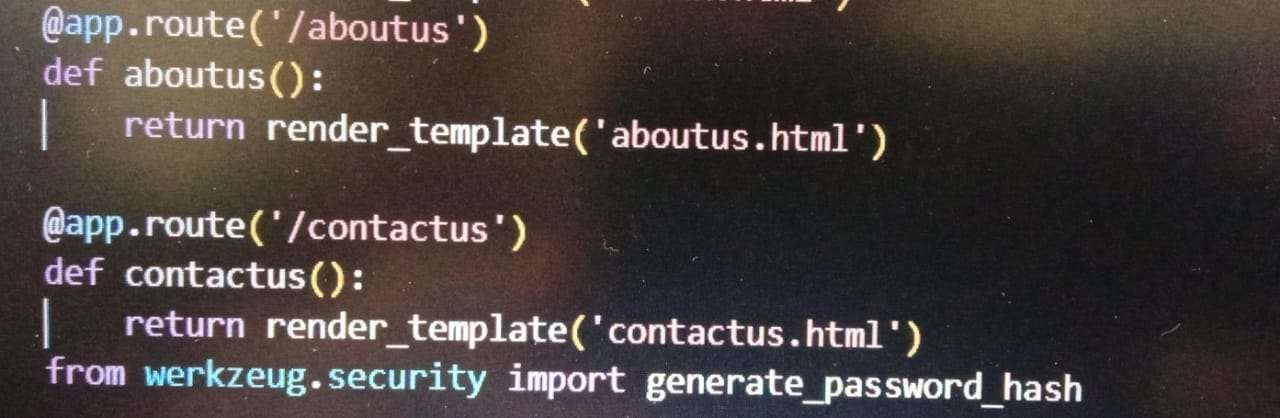
Description: de ine signup route to validate create account form ields, hash the user password using Bcrypt, store the new user in DynamoDB with a login count, and redirect to login page.

* login Route (GET/POST):

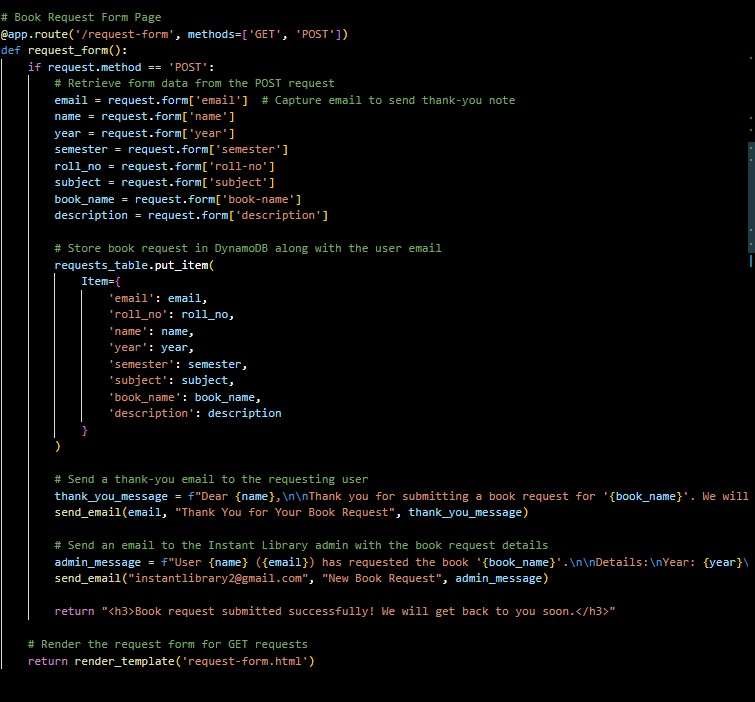


Description: de ine /login route to validate user credentials against DynamoDB, check the password using Bcrypt, update the login count on successful authentication, and redirect users to the patient dashboard or doctor dashboard based on button they have chosen.

Aboutus and contactus routes:



Description: de ine /aboutus and contactus page to render from the index page based on buttons to handle redirection to aboutus page and contactus page and /<aboutus>.html and <contactus>.html dynamic route to render from/to speci ic pages.

* Request Routes:
* 

Description: de ine /request-form route to capture book request details from users, store the request in DynamoDB, send a thank-you email to the user, notify the admin, and con irm submission with a success message.

Logout Route:



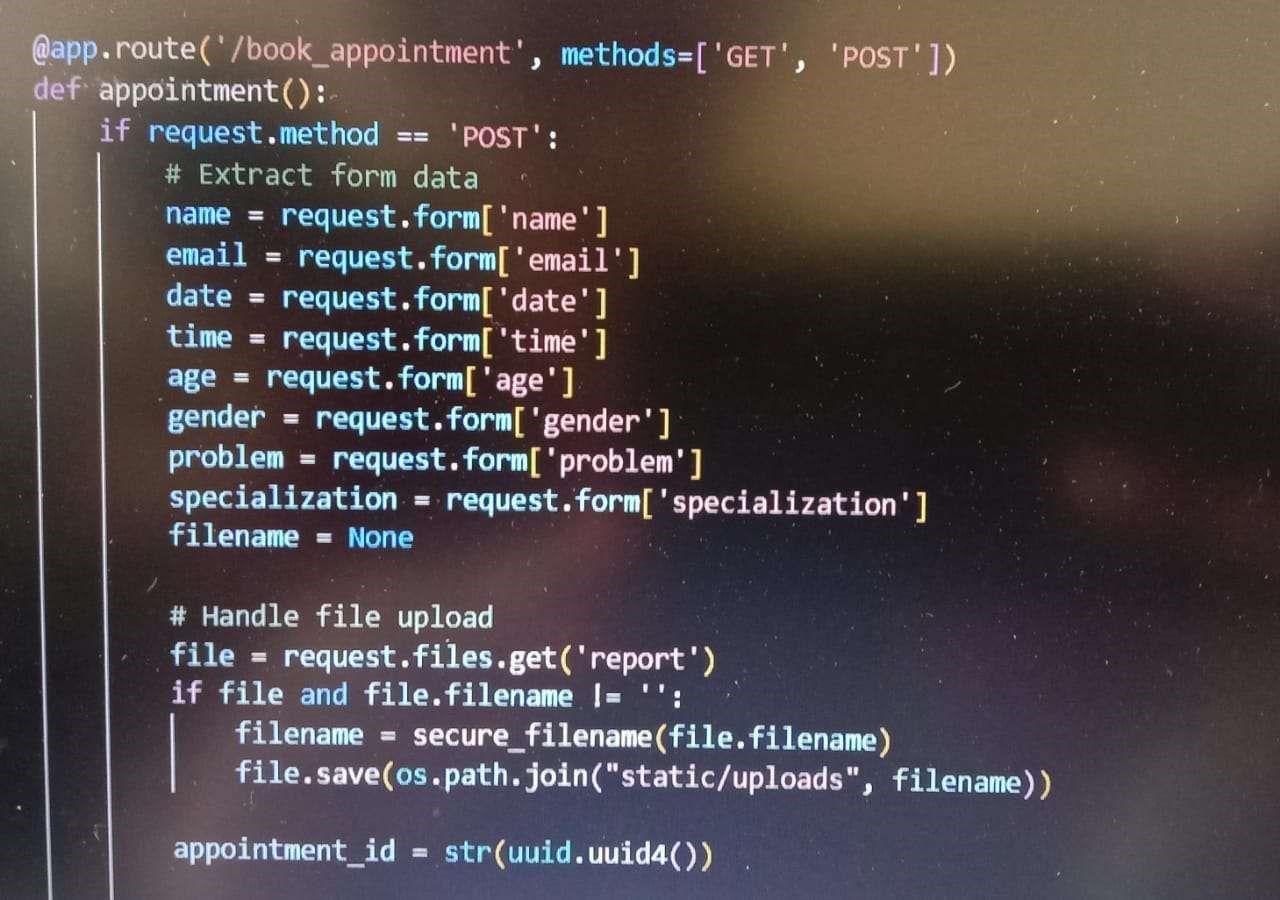
Description: de ine /exit route to render the exit.html page when the user chooses to leave or close the application.

Deployment Code:



Description: start the Flask server to listen on all network interfaces (0.0.0.0) at port 80 with debug mode enabled for development and testing.

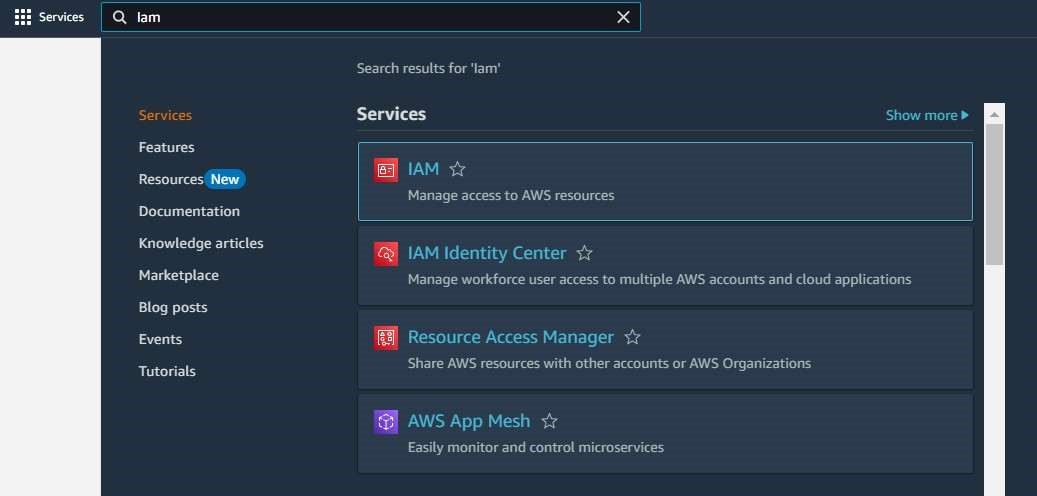
Book Appointment route:

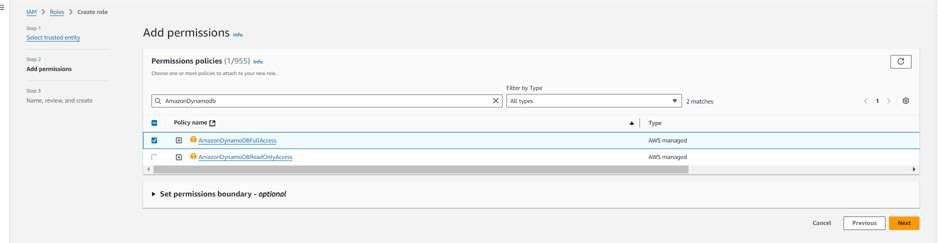
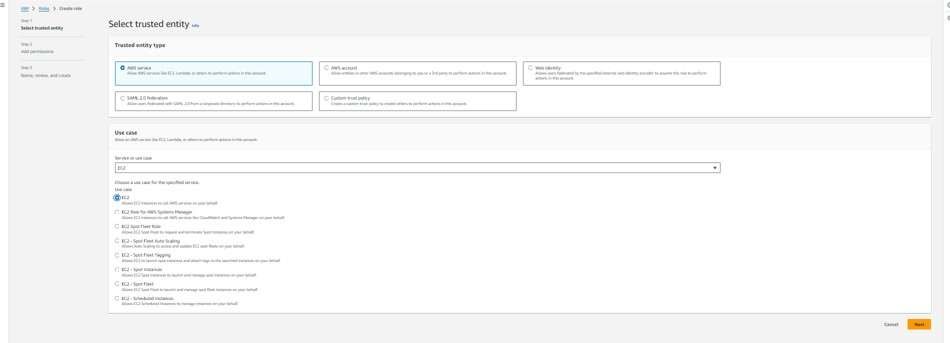
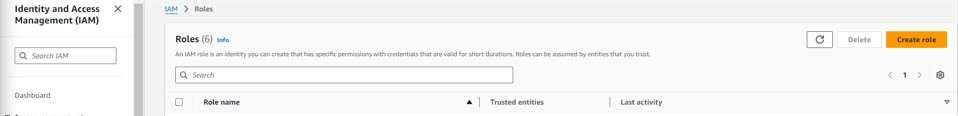


# Milestone 5: IAM Role Setup

* Activity 5.1:Create IAM Role.

○ In the AWS Console, go to IAM and create a new IAM Role for EC2 to interact with DynamoDB and SNS.

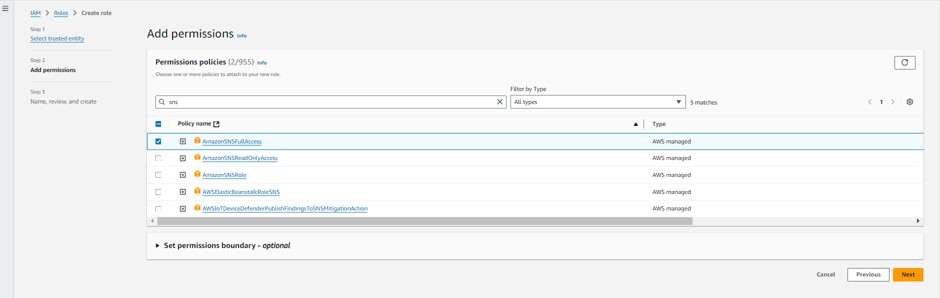


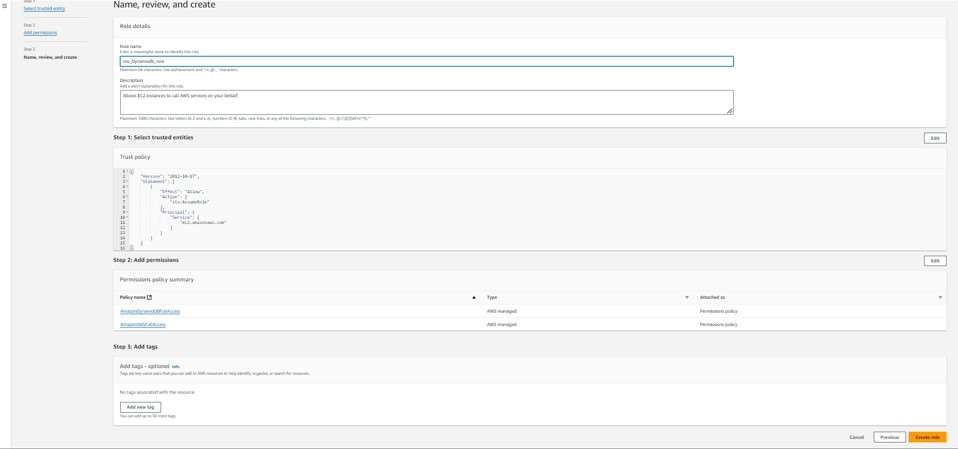


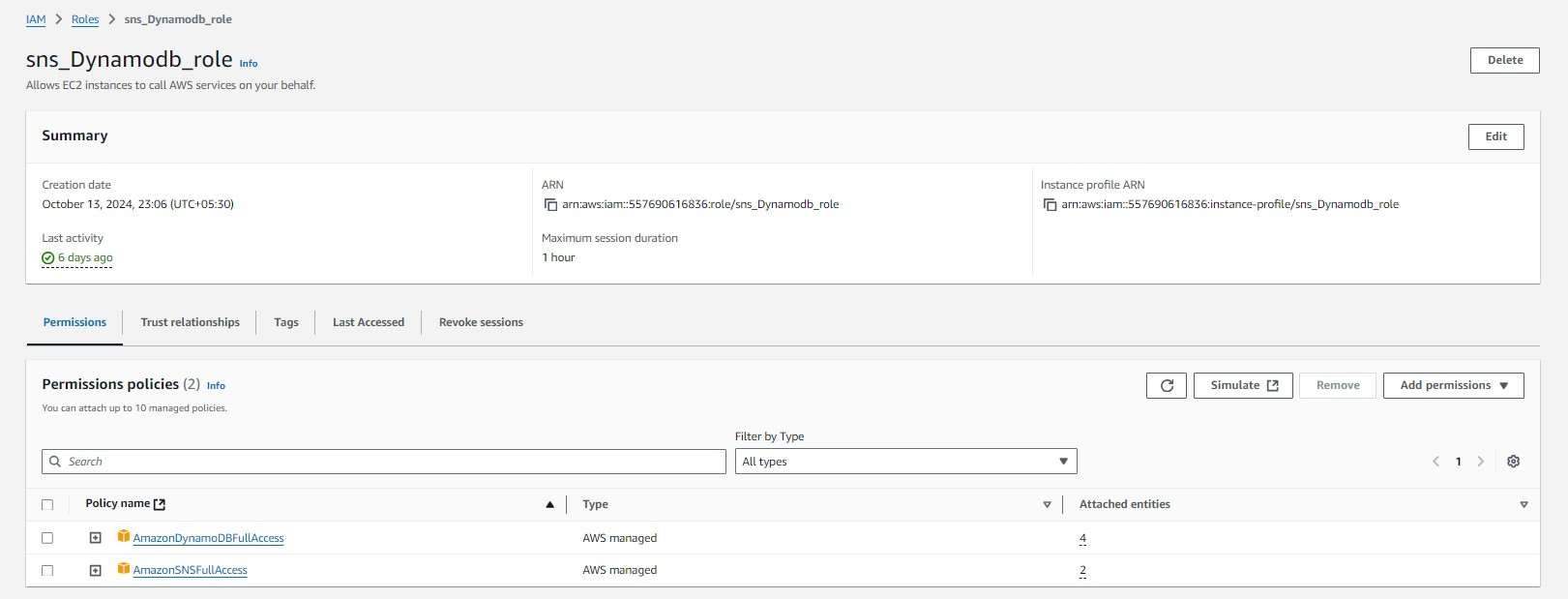
* Activity 5.2: Attach Policies.

Attach the following policies to the role:

* AmazonDynamoDBFullAccess: Allows EC2 to perform read/write operations on DynamoDB.
* AmazonSNSFullAccess: Grants EC2 the ability to send noti ications via SNS.



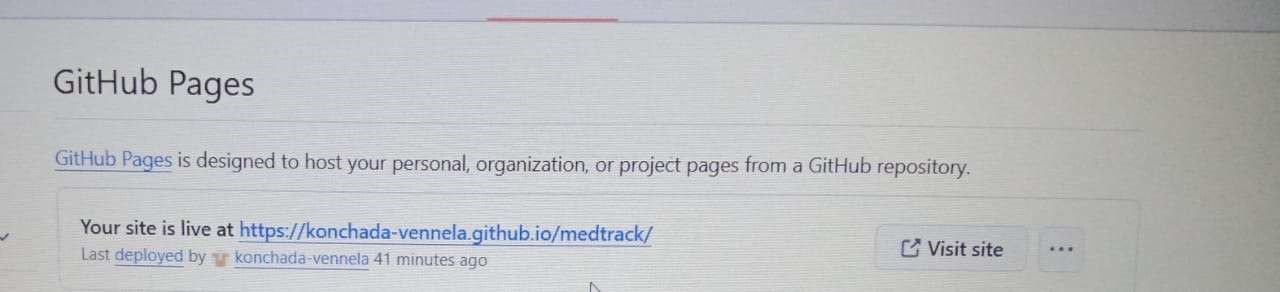




# Milestone 6: EC2 Instance Setup

* Note: Load your Flask app and Html iles into GitHub repository.

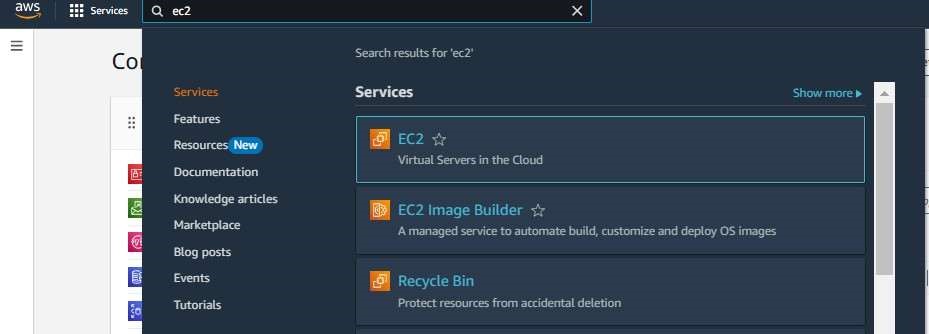




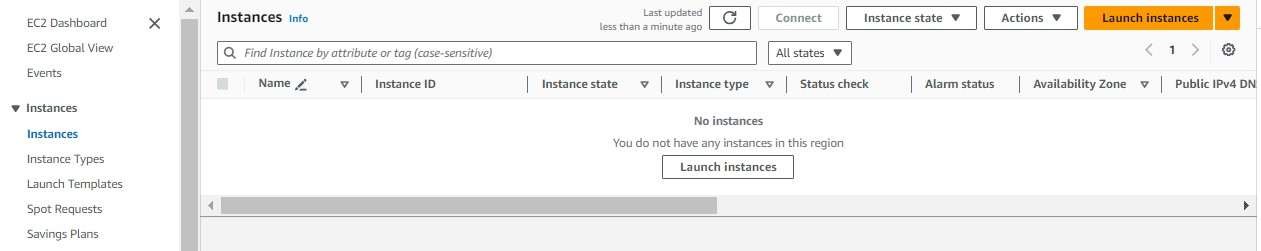
* Activity 6.1: Launch an EC2 instance to host the Flask application.

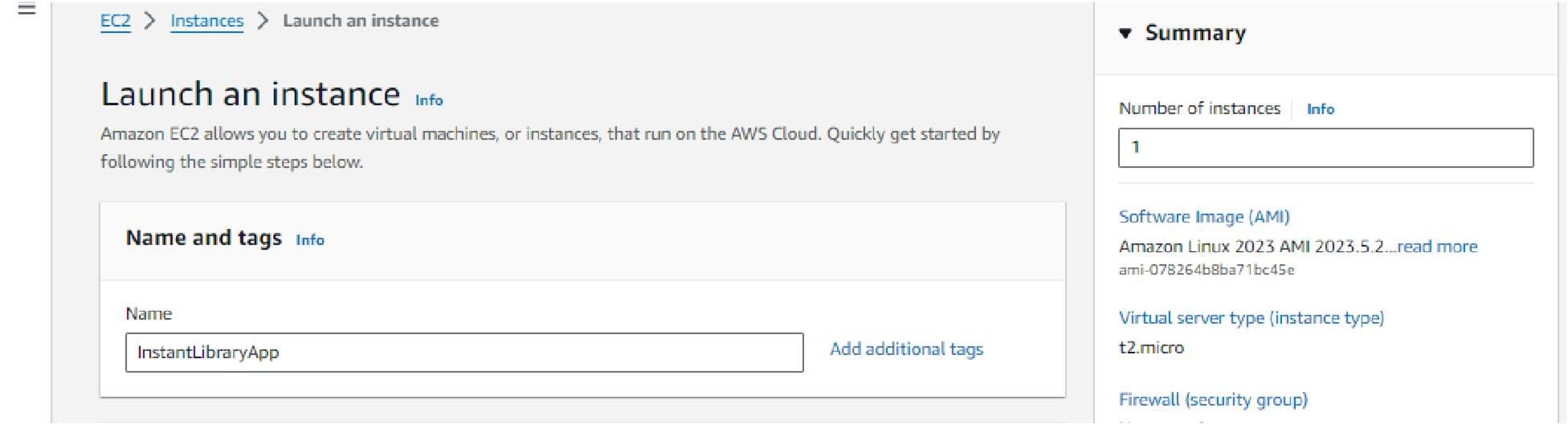
* Launch EC2 Instance

○ In the AWS Console, navigate to EC2 and launch a new instance.

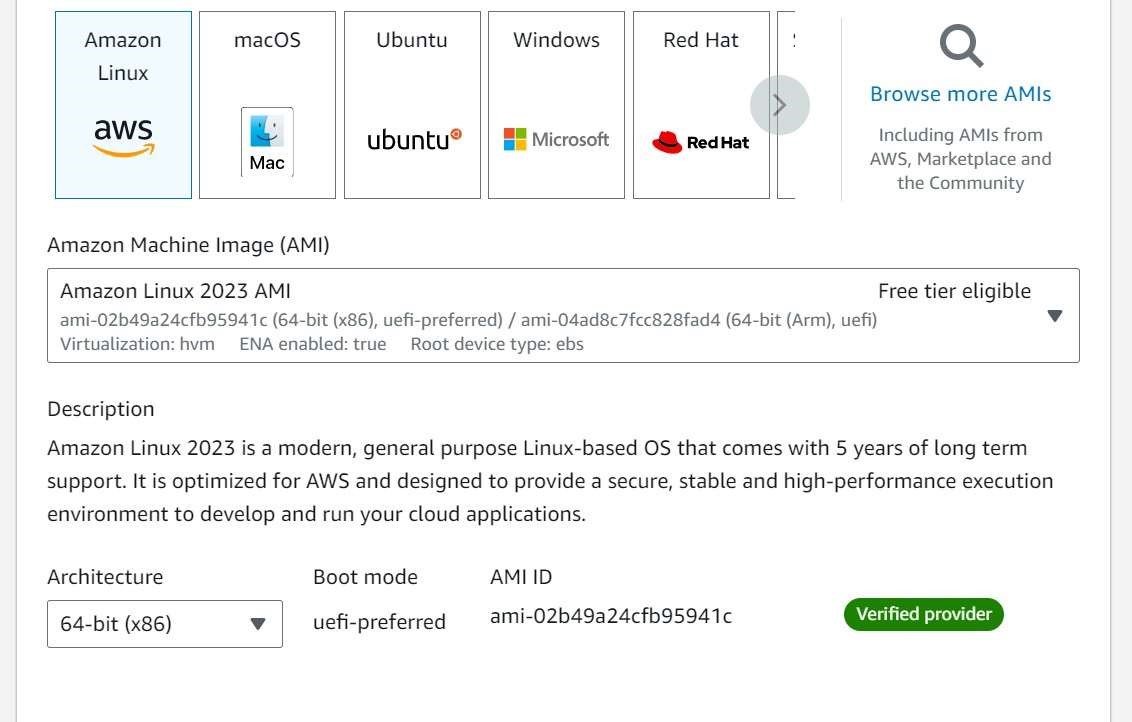


* Click on Launch instance to launch EC2 instance

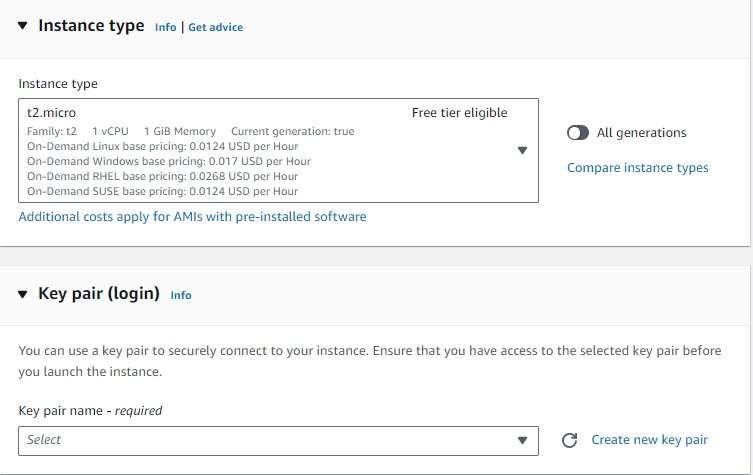




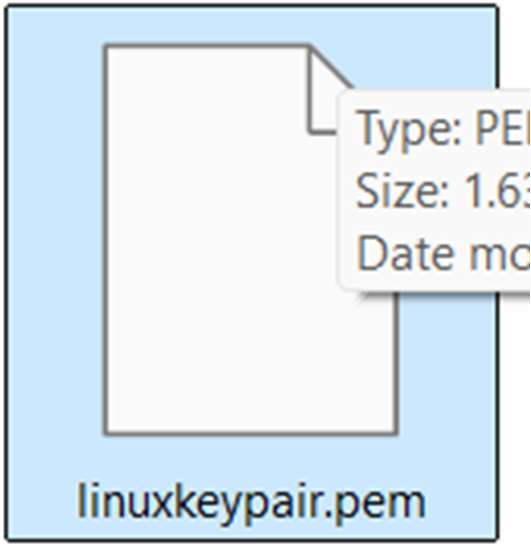
* Choose Amazon Linux 2 or Ubuntu as the AMI and t2.micro as the instance type (free-tier eligible).

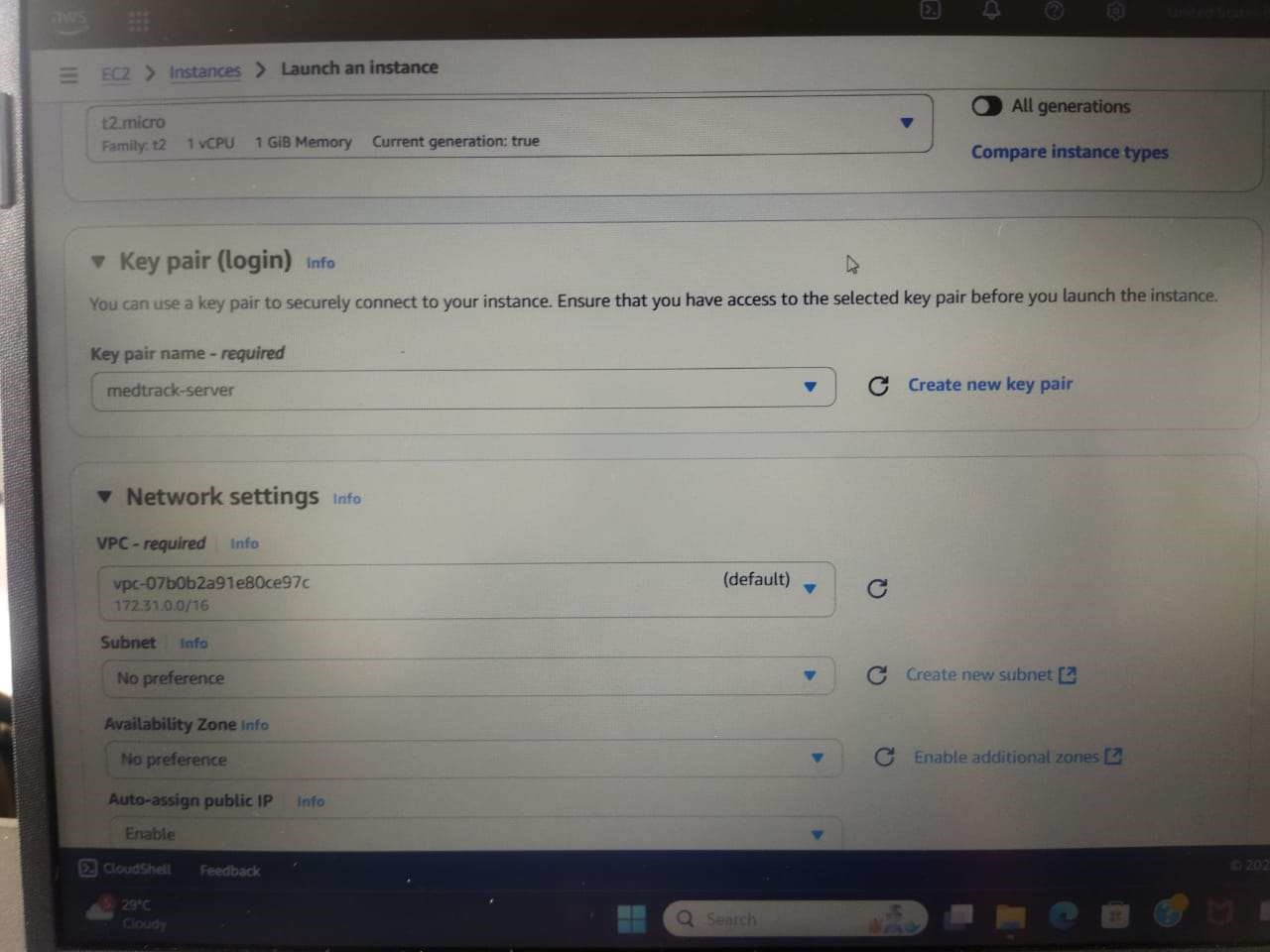


* Create and download the key pair for Server access.

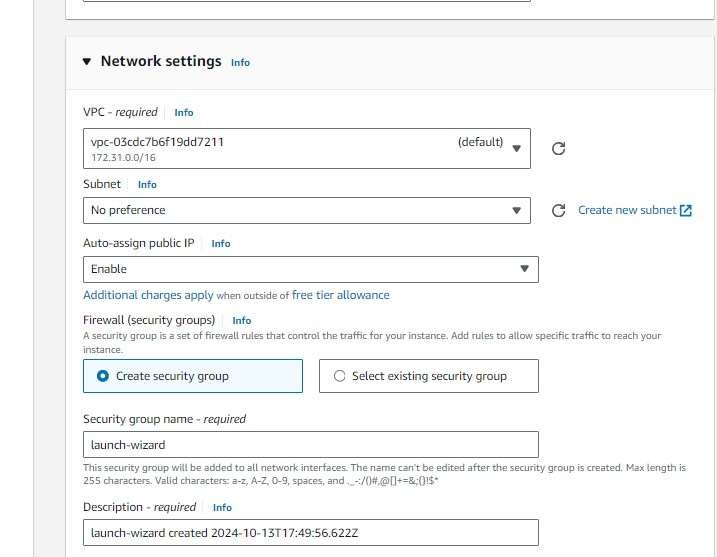


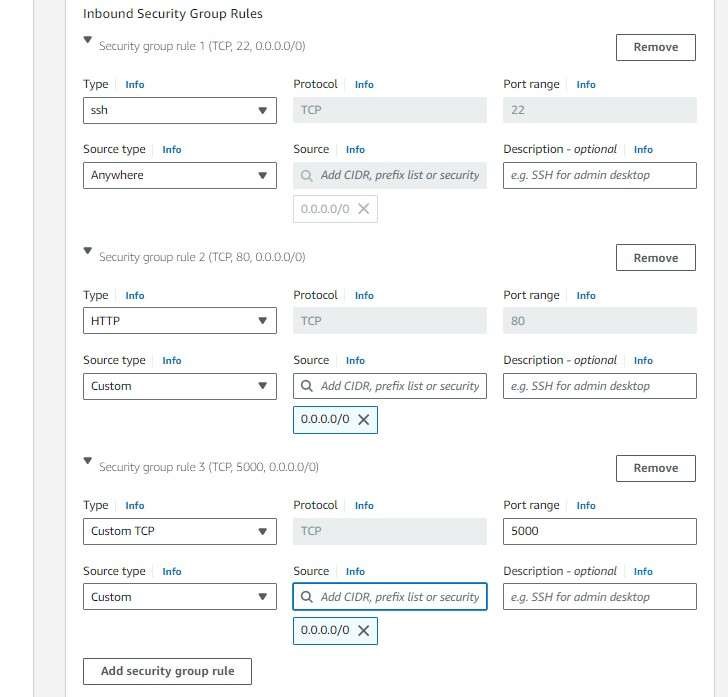






* Activity 6.2:Con igure security groups for HTTP, and SSH access.

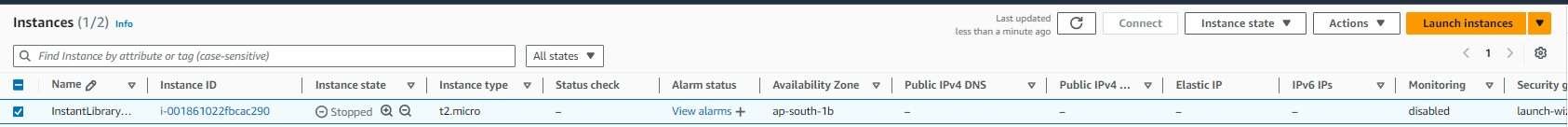




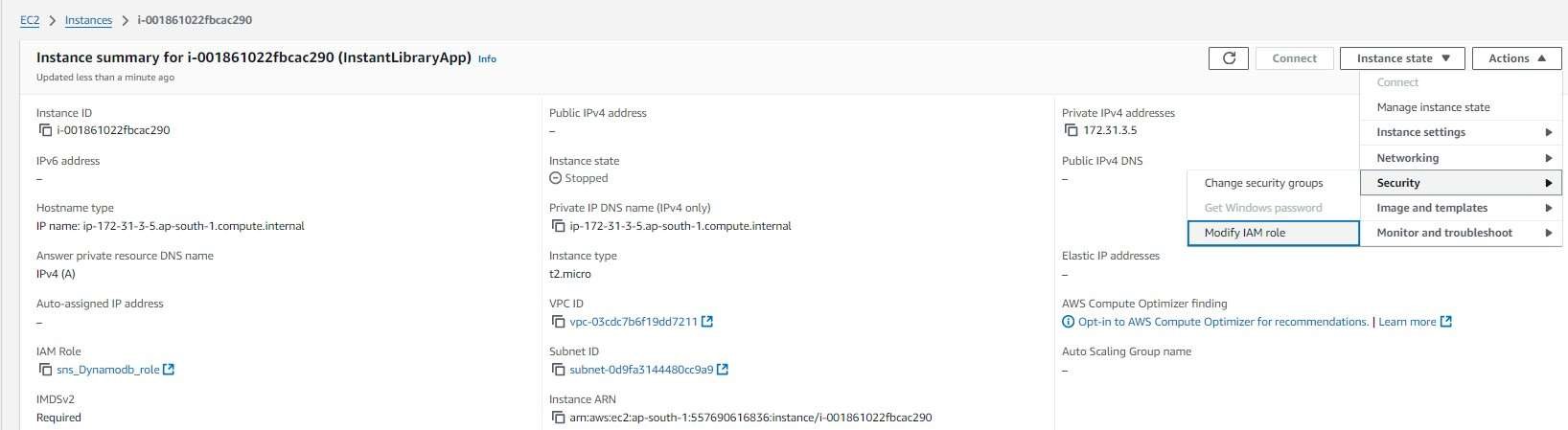


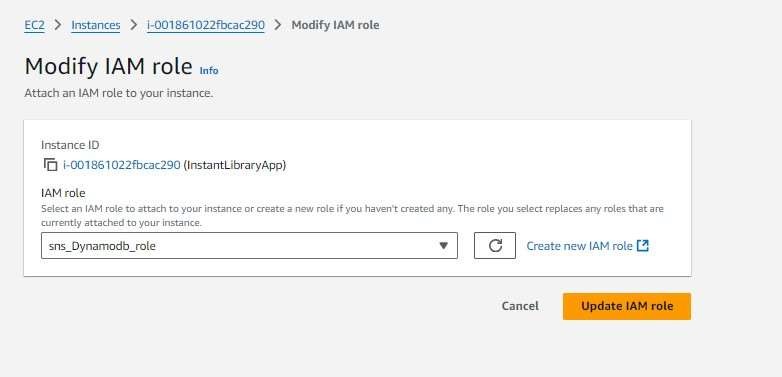
* To connect to EC2 using EC2 Instance Connect, start by ensuring that an IAM role is attached to your EC2 instance. You can do this by selecting your instance, clicking on Actions, then navigating to Security and selecting Modify IAM Role to attach the appropriate role. After the IAM role is connected, navigate to the EC2 section in the AWS Management Console. Select the EC2 instance you wish to connect to. At the top of the

EC2 Dashboard, click the Connect button. From the connection methods presented, choose EC2 Instance Connect. Finally, click Connect again, and a new browser-based terminal will open, allowing you to access your EC2 instance directly from your browser.

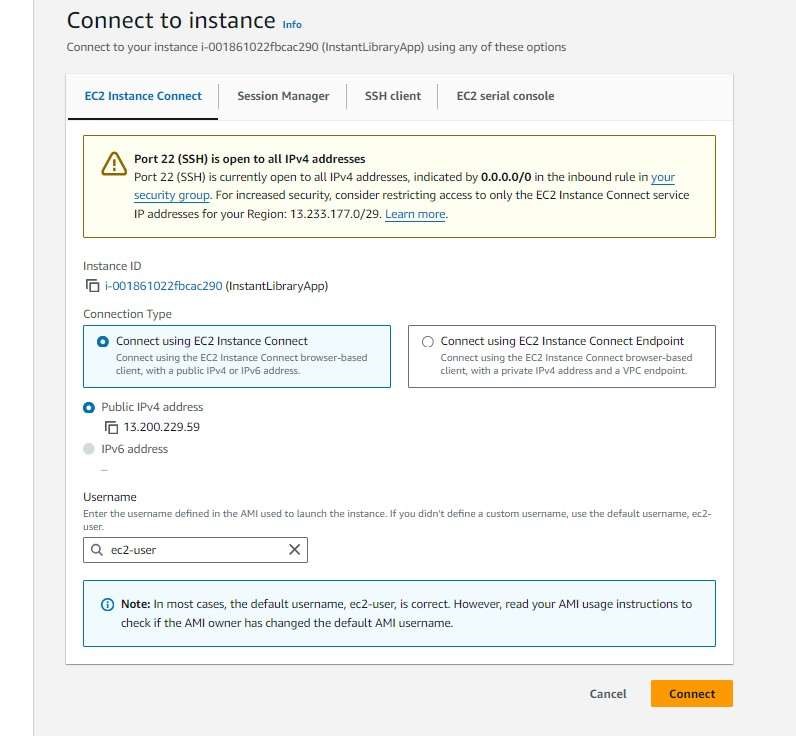


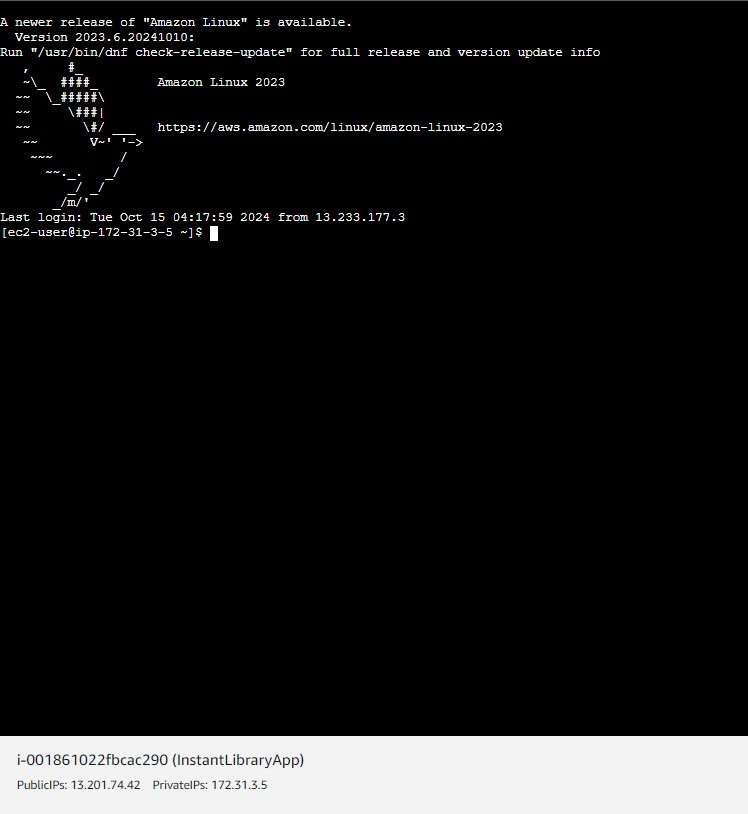






* Now connect the EC2 with the iles





Milestone 7: Deployment on EC2

Activity 7.1: Install Software on the EC2 Instance

Install Python3, Flask, and Git:

On Amazon Linux 2:

sudo yu update-y

sudo yum install python3 git sudo pip3 install lask boto3

Verify Installations:

lask --version

git --version

Activity 7.2:Clone Your Flask Project from GitHub

Clone your project repository from GitHub into the EC2 instance using Git.

Run: ‘https://github.com/konchada-vennela/medtrack.git’ Note: change your-githubusername and your-repository-name with your credentials here: ‘git clone

‘https://github.com/konchada-vennela/medtrack.git ‘

● This will download your project to the EC2 instance.

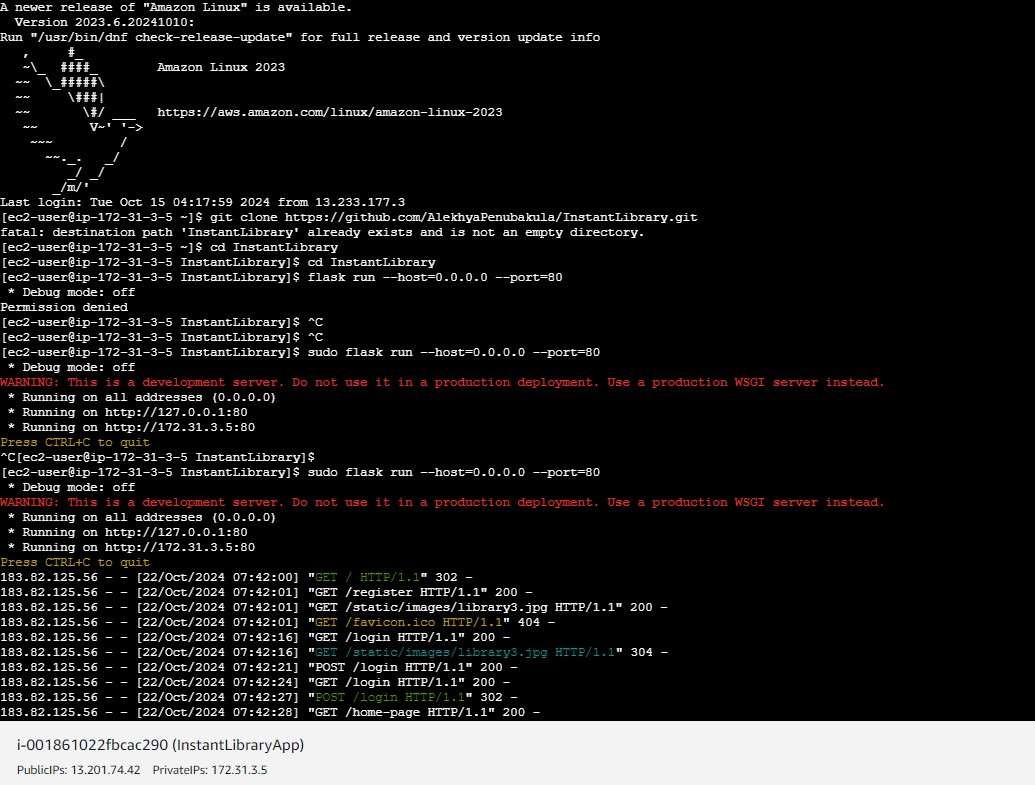
To navigate to the project directory, run the following command:

cd InstantLibrary

Once inside the project directory, con igure and run the Flask application by executing the following command with elevated privileges:

Run the Flask Application

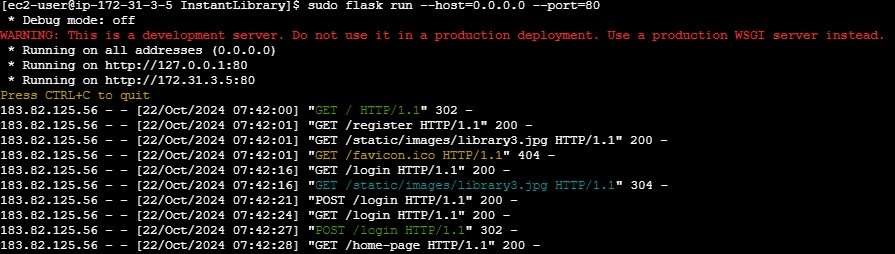
sudo lask run --host=0.0.0.0 --port=80



Verify the Flask app is running: http://your-ec2-public-

ip

○ Run the Flask app on the EC2 instance



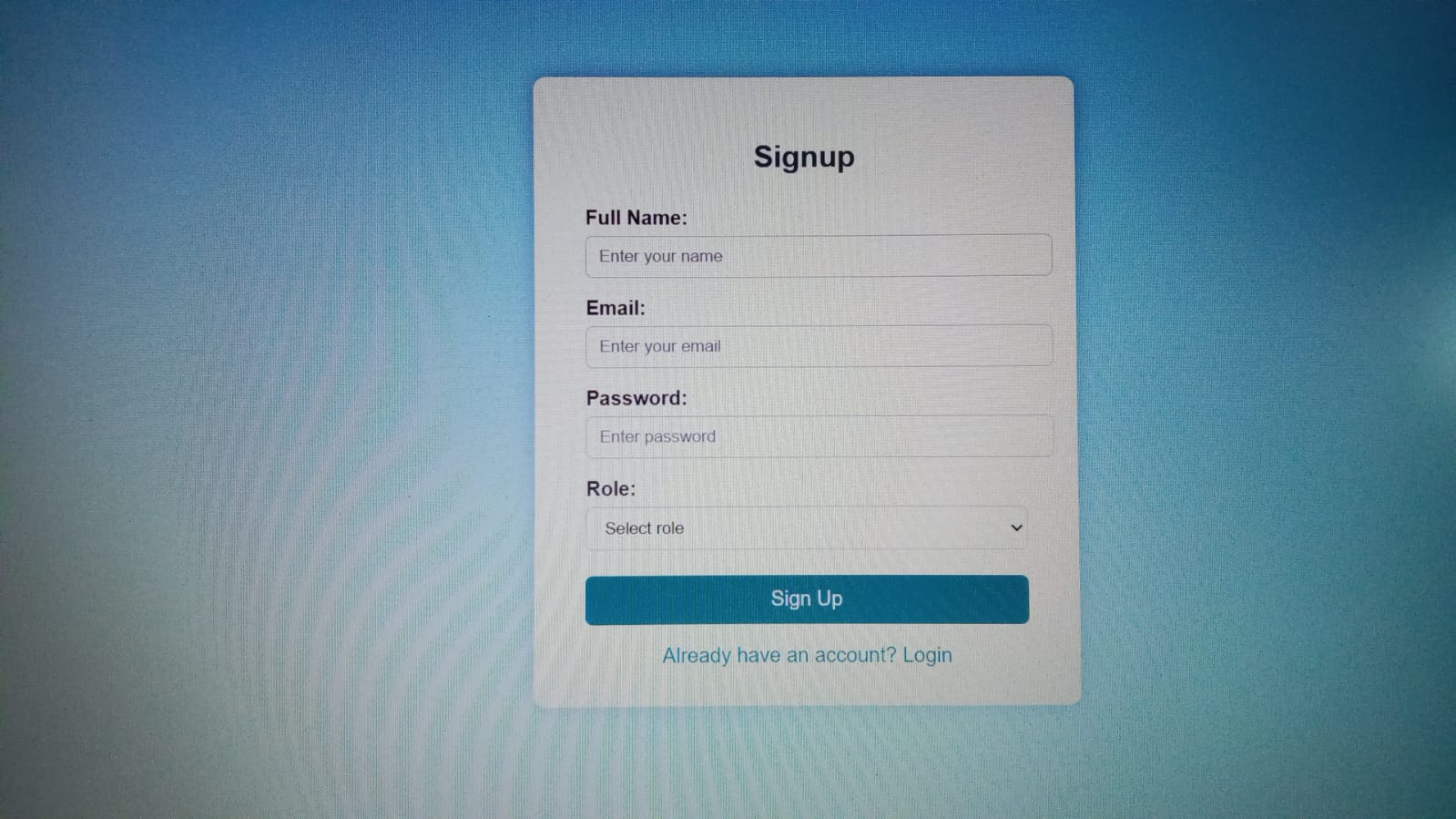
Access the website through:

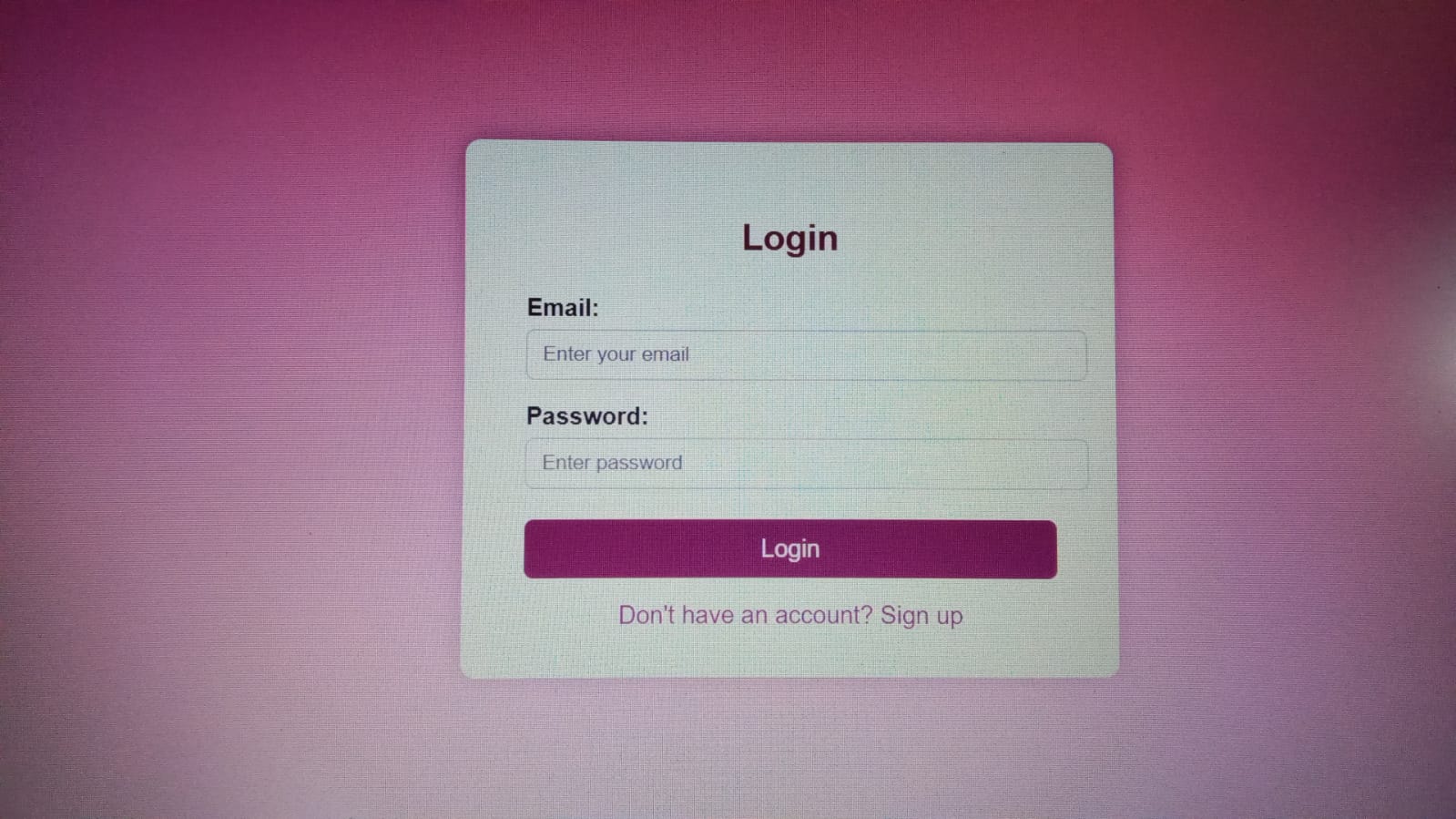
PublicIPs: https:// 3.85.104.16 /

## Milestone 8: Testing and Deployment

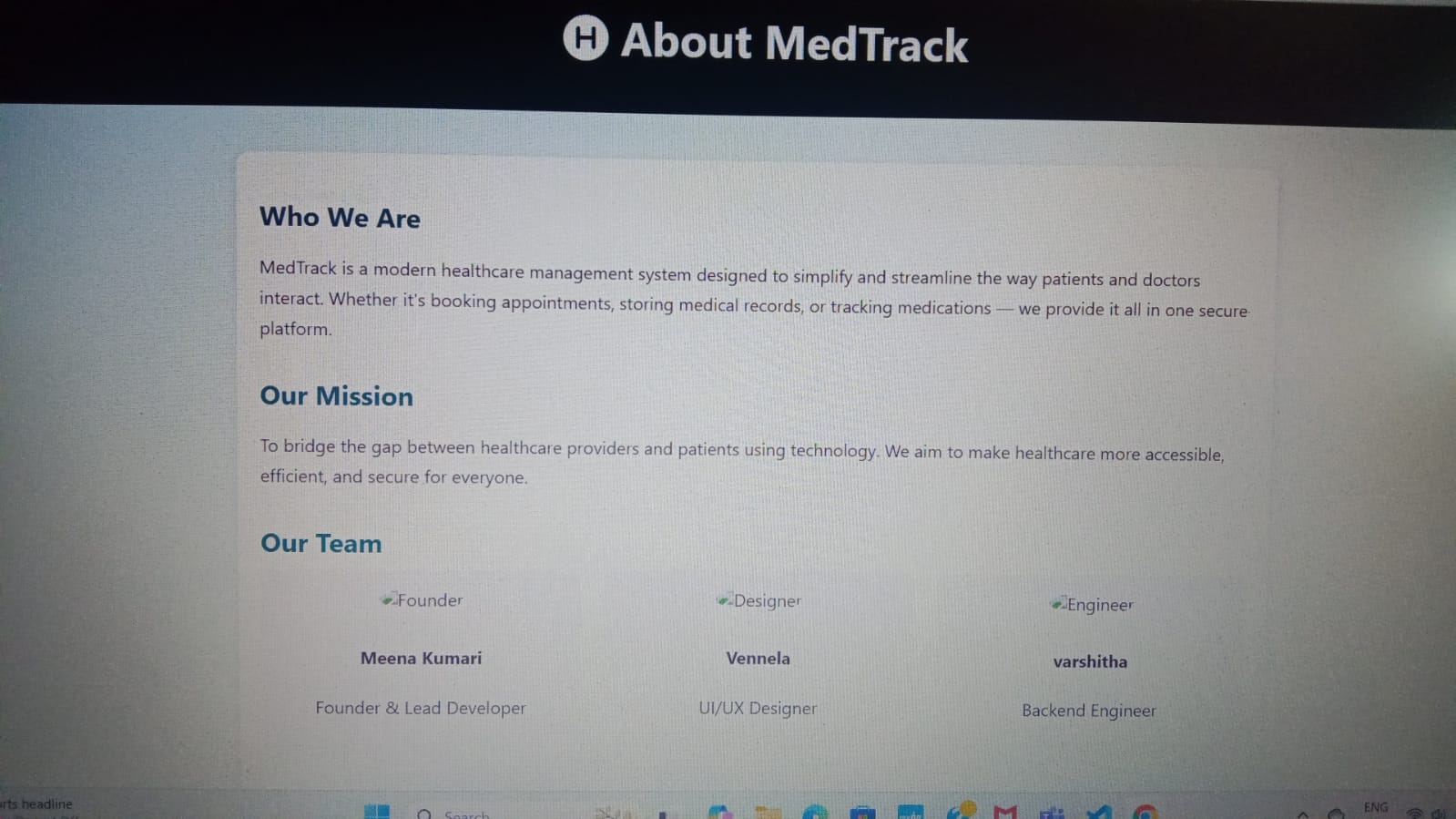
● Activity 8.1: Conduct functional testing to verify user registration, login, book requests, and notifications.

Register Page:

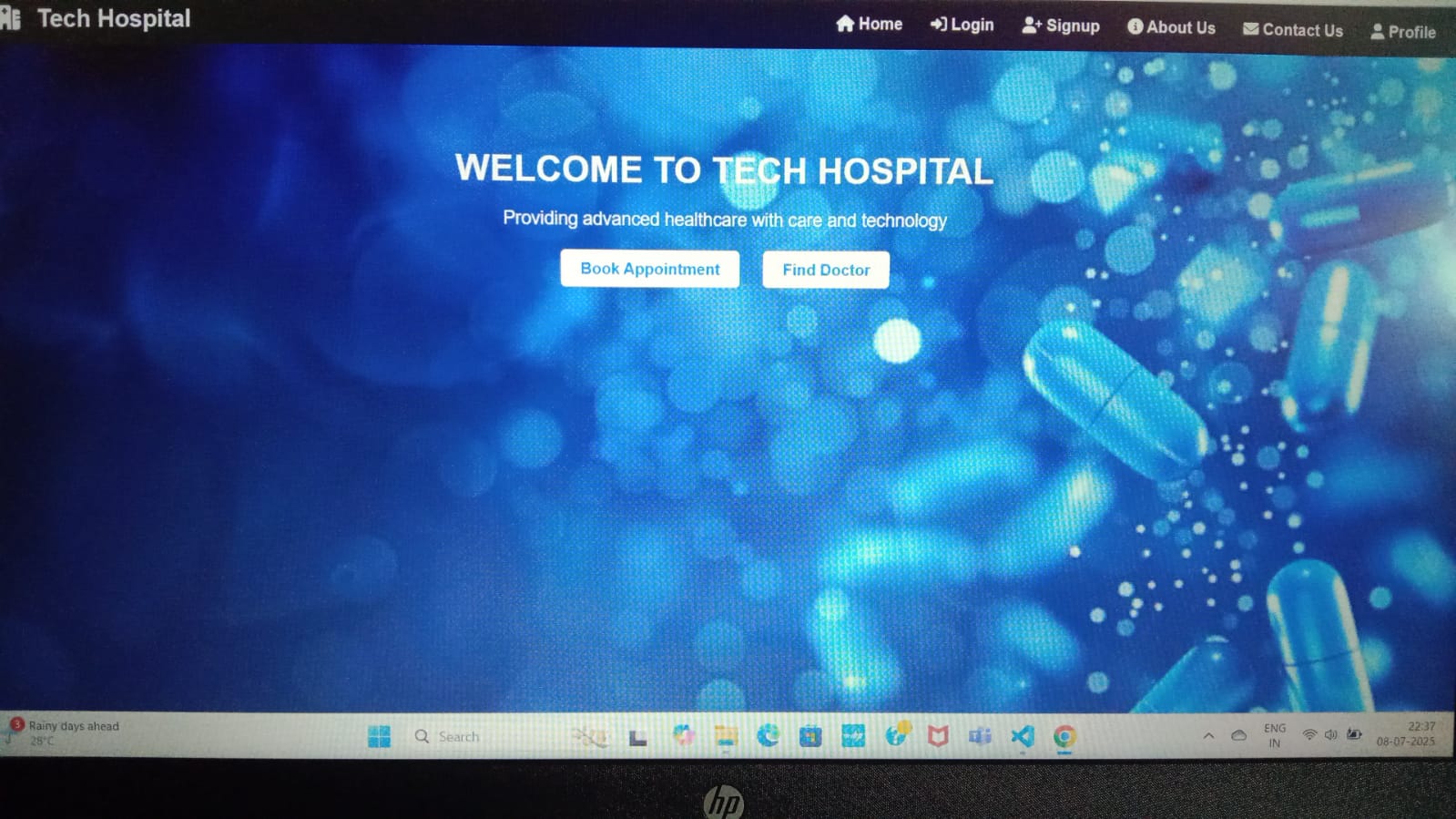


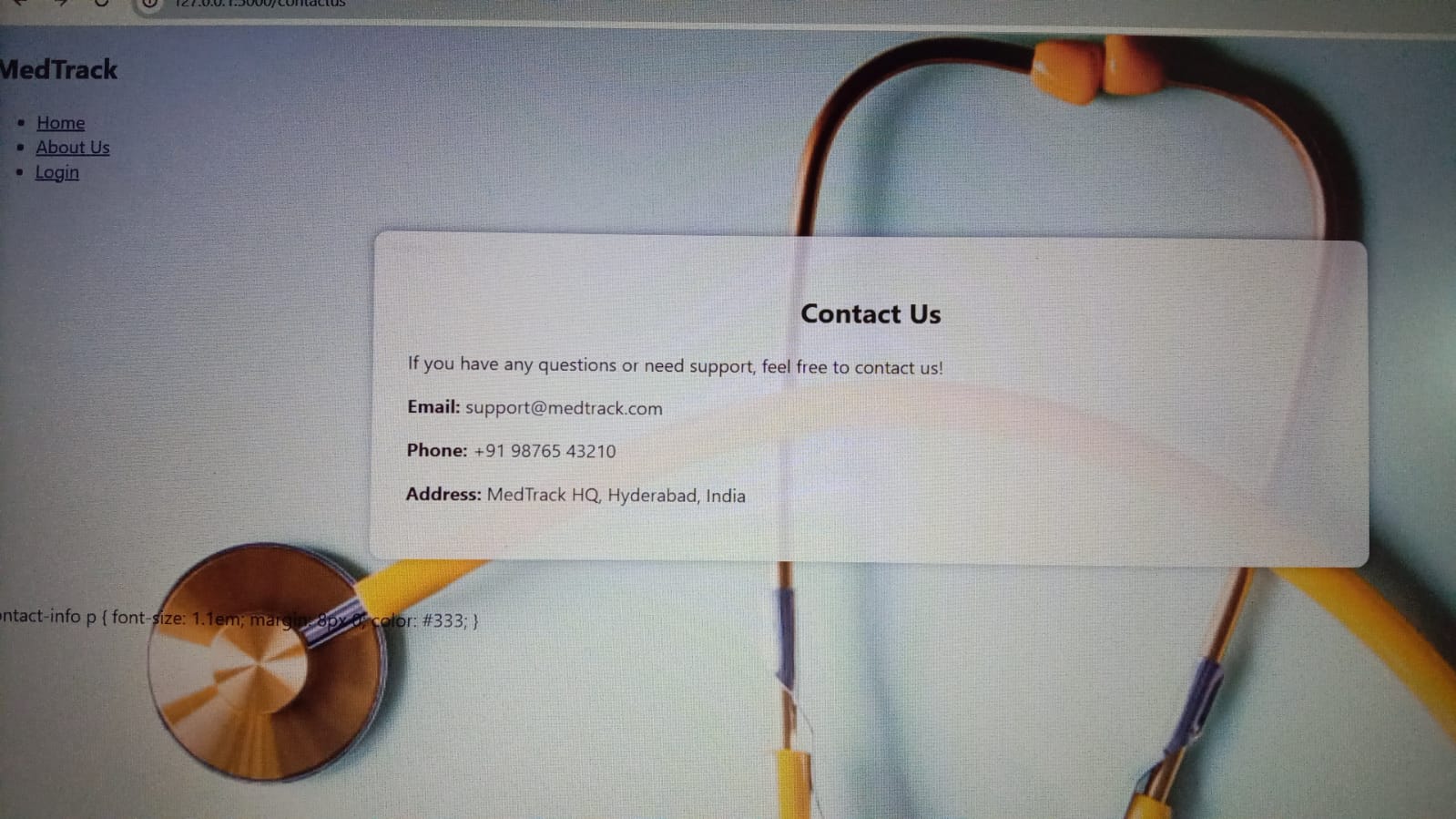
 Login Page:

About us page**:-**

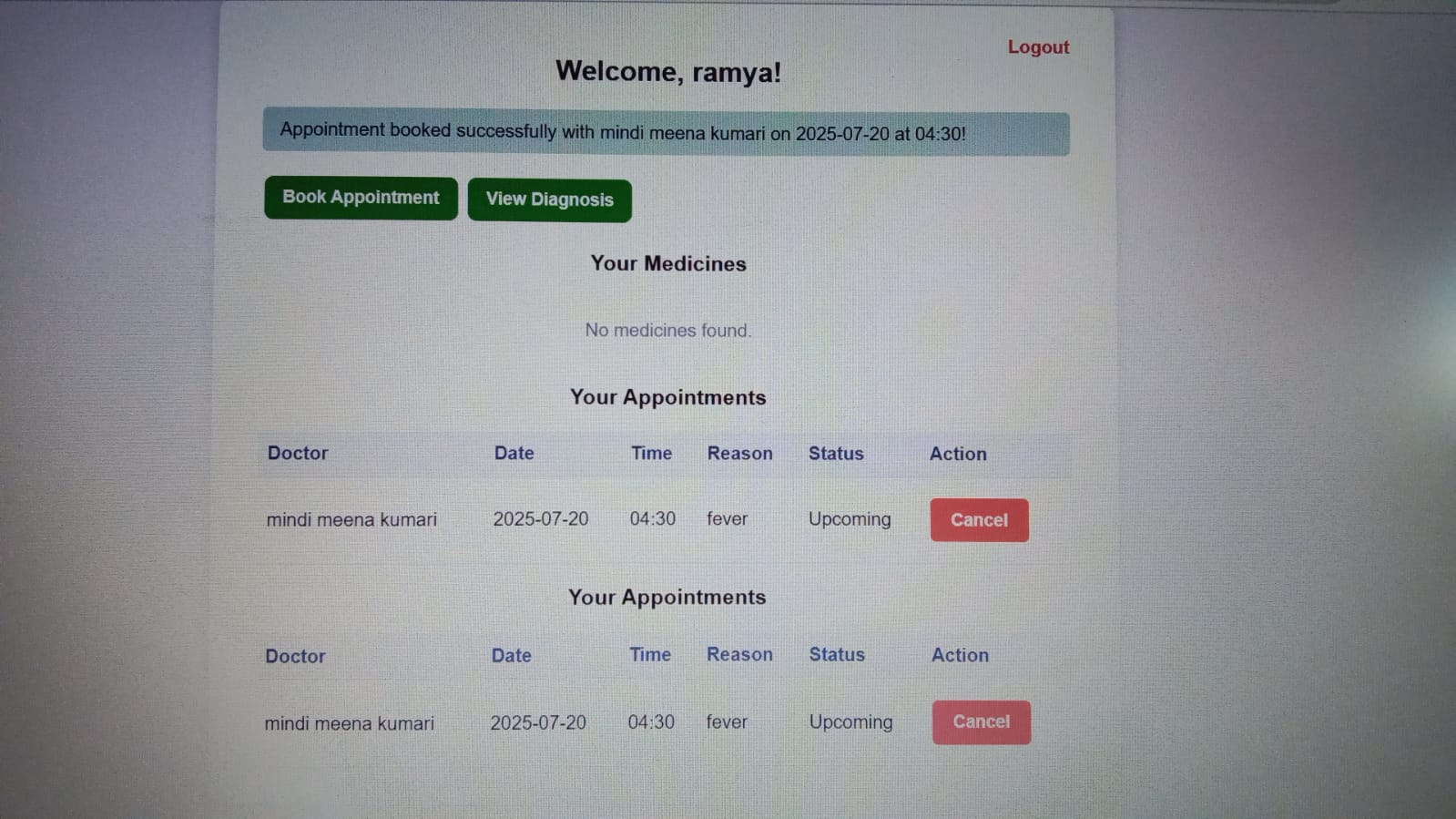


Home Page:-

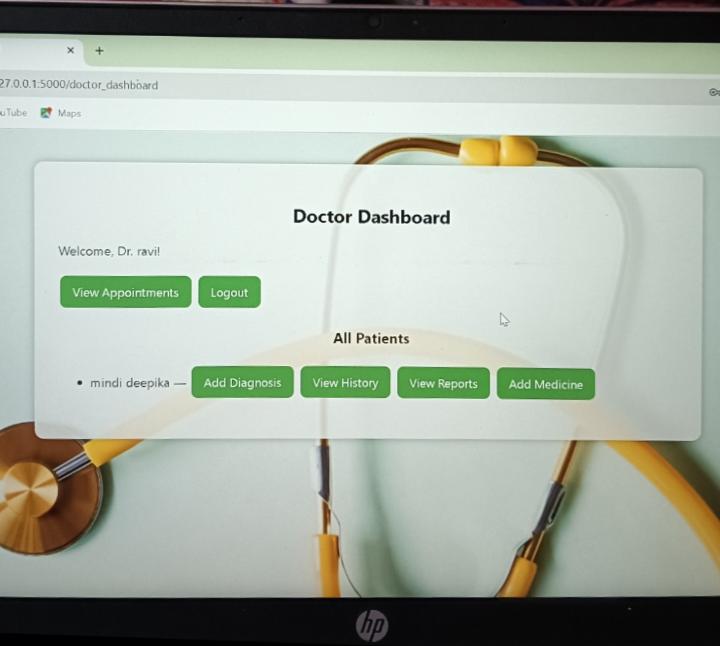


Contact Page: 

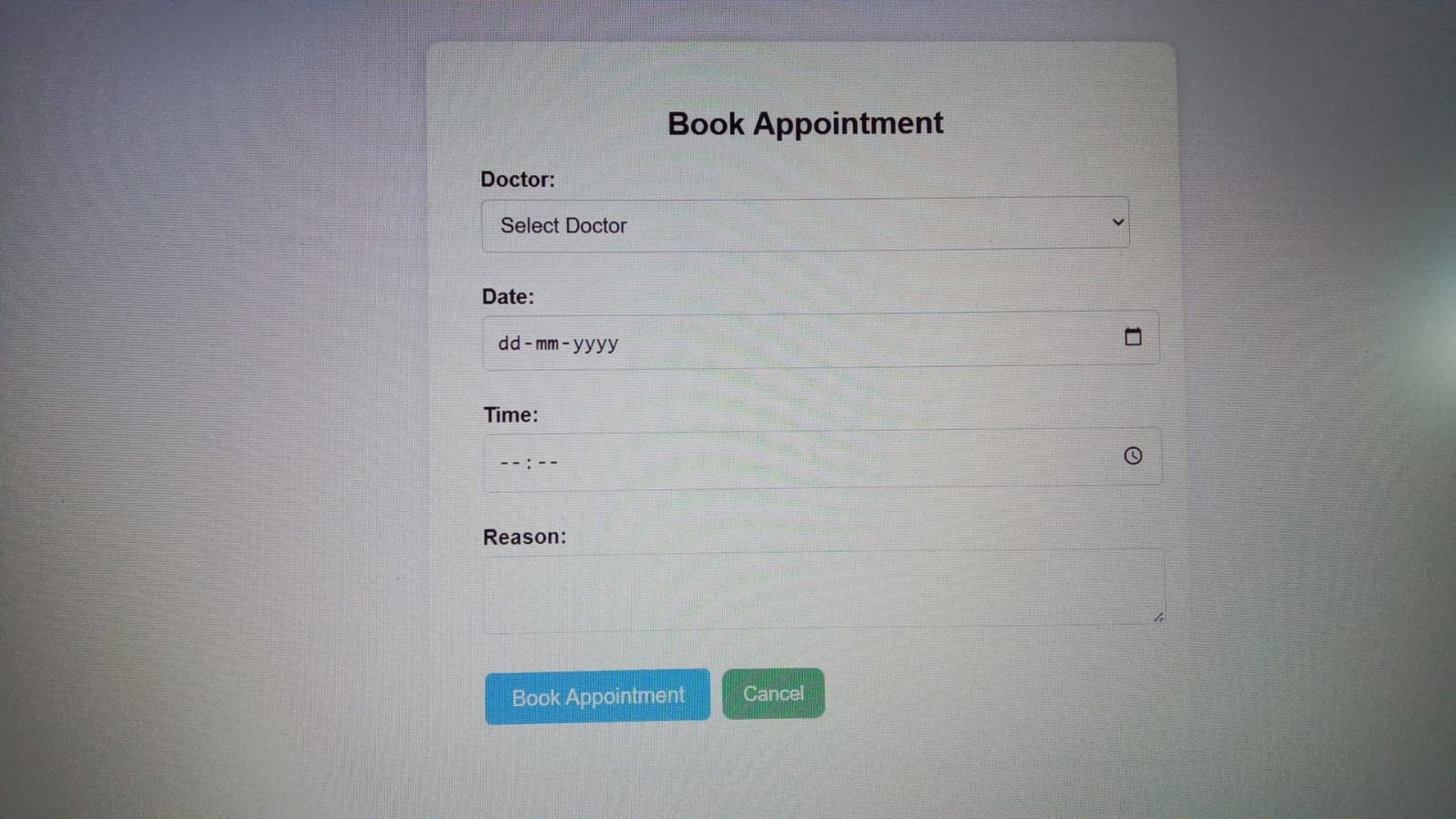
Patient-dashboard page:



Doctor-dashboard Page:



Book Appointment Page:



Add Medicine page: Upload Report page:



Add diagnosis page: Medical History page:

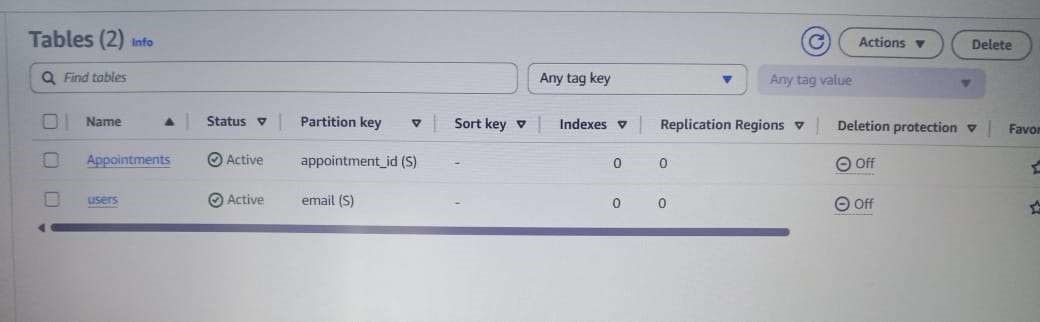


Exit:



Dynamodb Database Updations:

1. Users table and Appointments table :



Conclusion:

The MedTrack application has been successfully developed and deployed using a robust cloud-based architecture tailored for modern healthcare environments. Leveraging AWS services such as EC2 for hosting, DynamoDB for secure and scalable patient data management, and SNS for real-time alerts, the platform ensures reliable and ef icient access to essential medical tracking services. This system addresses critical challenges in healthcare such as managing patient records, monitoring medication schedules, and ensuring timely communication between healthcare providers and patients.

The cloud-native approach enables seamless scalability, allowing MedTrack to support increasing numbers of users and data without compromising performance or reliability. The integration of Flask with AWS ensures smooth backend operations, including patient registration, medication reminders, and health updates. Thorough testing has validated that all features—from user onboarding to alert noti ications—function reliably and securely.

In conclusion, the Med Track application delivers a smart, ef icient solution for modernizing healthcare management, improving patient care, and streamlining communication between medical staff and patients. This project highlights the transformative power of cloud-based technologies in solving real-world challenges in the healthcare sector.